VŠB – Technická univerzita Ostrava

Fakulta elektrotechniky a informatiky

Katedra kybernetiky a biomedicínského inženýrství

Detekce objektů v medicínských snímcích

Detection of object in medical images

2014 **Kristýna Hančarová**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra kybernetiky a biomedicínského inženýrství

Zadání bakalářské práce

Student: **Kristýna Hančarová**

Studijní program: B2649 Elektrotechnika

Studijní obor: 3901R039 Biomedicínský technik

Téma: Detekce objektů v medicínských snímcích
Detection of Objects in Medical Images

Zásady pro vypracování:

1. Seznámení se s metodami, které se používají pro segmentaci medicínských obrazových dat.
2. Seznámení se s matematickými metodami pro detekci hran v MALTABu.
3. Seznámení se s dvourozměrnou konvolucí a její aplikací ve zpracování obrazu.
4. Návrh a implemetace dostupných funkcí MATLABu pro realizaci detektorů hran.
5. Návrh a implementace vlastních funkcí pro detekci hran.
6. Porovnání účinnosti detekce objektů v medicínských snímcích.
7. Návrh a implementace filtrů na základě konvoluce pro obrazová data.
8. Vytvoření graficko uživatelského rozhraní v MATLABu pro detekci objektů v medicínských snímcích.
9. Zhodnocení dosažených výsledků práce.

Seznam doporučené odborné literatury:

[1] SOJKA, Eduard. *Digitální zpracování a analýza obrazu.* Skripta. 1. vyd. Ostrava: VŠB - Technická univerzita, 2000, 133 s. ISBN 80-7078-746-5.
[2] HLAVÁČ, Václav a Milan ŠONKA. *Počítačové vidění.* Praha: Grada, 1992, 272 s. ISBN 80-85424-67-3.
[3] RAJMIC, Pavel. *Základy počítačové sazby a grafiky.* Brno: Vysoké učení technické v Brně, 2012. s. 160. ISBN 978-80-214-4451-5.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Jan Kubíček**

Datum zadání: 01.09.2013
Datum odevzdání: 07.05.2014

doc. Ing. Jiří Koziorek, Ph.D.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

Datum odevzdání: 7. 5. 2014

Kristýna Hančarová

Poděkování

Mé velké díky patří Ing. Janu Kubíčkovi za odborné vedení bakalářské práce. Především bych chtěla poděkovat mé rodině a přátelům za bezednou trpělivost a obrovskou podporu nejen v době realizace této práce, ale také po celou dobu studií.

Abstrakt

Cílem práce je seznámit se s problematikou segmentace obrazu. Po teoretickém rozboru je tato problematika následně zpracována v softwarovém prostředí MATLAB (verze 2011a). Mezi hlavní části zpracování obrazu patří detekce hran, proto je práce zaměřena především na hranové operátory, a to využívající aproximaci derivace prvního řádu, druhého řádu a operátory aproximující obrazovou funkci. Práce se dále zabývá konvolucí, která je nezbytnou součástí při segmentaci obrazu. Obsahuje také návrh a realizaci implementace dostupných algoritmů v MATLABu pro detekci hran obrazu, a tím napomáhá uživateli snadněji rozpoznat zájmové objekty na určitých medicínských snímcích. Stěžejním úkolem práce je rozšířit část algoritmů o vlastní implementaci hranových detektorů pro nalezení hledaných objektů v obraze. Práce také obsahuje praktický výstup zpracování medicínských obrazových dat, konkrétněji způsob detekce cévního řečiště a zrakového nervu pomocí Kirschova operátoru.

Klíčová slova

segmentace, zpracování obrazu, MATLAB, hranové detektory, konvoluce, GUI rozhraní.

Abstract

The aim of this bachelor thesis is image segmentation. The theoretical analysis is consequently followed by its processing in software environment MATLAB (version 2011a). One of the main parts of image processing is the edge detection and therefore this work focuses especially on edge detection operators, namely those using approximation of the derivative of the first order, second order and operators approximating image function. The thesis deals with convolution which forms a necessary part image segmentation. It also contains the suggestion and realization of available algorithms implementation in MATLAB for the image edge detection and hereby helps the user to recognize objects of interest in individual medical images easily. The principal objective of this work is to implement own edge detection operators into a part of algorithms in order to find individual objects in images. The thesis also includes practical output of medical image data processing, specifically a method of detecting blood vessel and optic nerveusing by Kirsch operator.

Key Words

segmentation, image processing, MATLAB, edge detectors, convolution, GUI interface.

Seznam použitých symbolů a zkratek

2D (Two dimension) – dvourozměrný

A_i – podmnožiny jasových úrovní

$D(f)$ – definiční obor

DoG – (difference of Gaussians) – diference Gaussiánů

$DoG(x, y)$ – výsledek diference Gaussiánu

GUI – (Graphical User interface) – grafické uživatelské prostředí

$G(x, y)$ – Gaussovo rozložení

$H(f)$ – obor hodnot

$I(x, y)$ – výstupní obraz

LoG – (Laplacian of Gaussian) – Laplacián Gaussiánu

R – rovinná oblast

T – konstanta práhu

f_{max} – maximální frekvence signálu

f_s – vzorkovací frekvence

$f(x, y)$ – obrazová funkce

$g(x, y)$ – binární obraz

h_i – konvoluční masky

$h(x, y)$ – konvoluční jádro na pozici x a y

k – interval udávající počet úrovní jasu

n – bitová šířka

x_a – maximální souřadnice vertikální osy obrazu

y_b – maximální souřadnice horizontální osy obrazu

ε – přesnost

σ – směrodatná odchylka

$\varphi(x, y)$ – směr gradientu

$\psi(x, y)$ – směr hrany

$|\nabla f(x, y)|$ – velikost gradientu

$|\nabla f_x(x, y)|$ – velikost gradientu pro horizontální detektor

$|\nabla f_{xx}(x, y)|$ – diference dvou bodů oddělených jedním pixelem pro horizontální osu

$\left|\nabla f_y(x, y)\right|$ – velikost gradientu pro vertikální detektor

$\left|\nabla f_{yy}(x, y)\right|$ – diference dvou bodů oddělených jedním pixelem pro vertikální osu

∇^2 – Laplaceův operátor

$|\nabla^2 f(x, y)|$ – velikost gradientu pomocí druhé derivace

$\nabla^2 G(x, y)$ – LoG operátor

Obsah

Úvod

V současnosti jde vývoj vědy a techniky obrovskými kroky kupředu, a to i v oblasti digitálního zpracování a analýzy obrazů. Zpracování obrazu je velice zajímavé téma a problematika kolem něj je značně rozsáhlá. S touto oblastí je úzce spjato počítačové vidění, které získává informace ze zachyceného obrazu a poskytuje náhled do lidského vidění. Díky počítačovému vidění dokážeme vnímat a vidět věci z úplně opačného úhlu pohledu, což nám může pomoci nejen v životě, ale také v lékařství, na které se zaměřuje například disciplína zabývající se biomedicínským zobrazováním (2D, 3D, 4D). Je to moderní, nadmíru rozvíjející se obor k překonání mezifáze mezi medicínou a inženýrstvím, rovněž jako věda a výpočetní technika.

Obrazy mohou vzniknout různými způsoby; pohledy z kamer, snímkováním, vícerozměrnými daty z různých lékařských vyšetření, a podobně. Práce je zaměřena na základní způsoby segmentace medicínských snímků, převážně z magnetické rezonance, a jsou zde využívány metody pro detekci hran v obrazech.

Segmentace obrazu je velmi rozsáhlá metoda z oblasti digitálního zpracování obrazu. Díky segmentaci dokážeme od sebe oddělit dílčí oblasti obrazu se společným charakterem. Cílem je identifikovat popředí a zdůraznit podstatné prvky obrazu, a metoda detekce hran je pouze částí, i když významnou, pro úpravu a správné zpracování obrazu.

Práce se zabývá teoretickými základy digitálního zpracování obrazu a jejich následnou aplikací při řešení daného tématu v prostředí MATLAB. Klíčovým nástrojem k řešení dané problematiky je jedna z metod digitálního zpracování obrazu, konkrétně pak detekce hran.

Text bakalářské práce je rozčleněn na jednotlivé části, z jejichž následného spojení a praktického použití získáme přehled o dané problematice. První část se týká teorie principu vzniku obrazu, jeho digitalizaci a zobrazení. Poté je část věnována již zmíněné segmentaci obrazu a prahování. Dalším z témat je konvoluce a hranové detektory, které jsou rozděleny podle základních idejí principu použití. Následně jsou podrobně rozebrány jednotlivé hranové detektory, které jsou dostupné v MATLABu. Hlavní a závěrečnou fází je vytvoření grafického uživatelského rozhraní GUI pro konvoluci a detekci hran a implementaci vlastních algoritmů pro detekci hran v obraze.

Pro přehlednost a pro oddělení dílčích oblastí je vytvořeno hlavní uživatelské rozhraní, které se následně větví na více aplikací. Zde má uživatel možnost použít jednotlivé funkce na medicínské snímky. Jako praktickou ukázku možnosti využití detekce hran v biomedicínské technice, konkrétně pak v očním lékařství, jsou použity snímky sítnice, kde je pomocí vlastního a Kirschova detektoru detekováno cévní řečiště a zrakový nerv.

1. VZNIK OBRAZU

Prvním krokem zpracování obrazu je snímání a následná digitalizace. Snímáním získáme obraz z reálného prostředí, například z fotoaparátu či kamery. Rozlišujeme dva typy obrazů, vektorový a rastrový. V našem případě se pohybujeme v oblasti rastru, jehož obraz je tvořen pixely.

Obrazovou funkci lze rozdělit na spojitou a diskrétní. Spojitá funkce má definiční obor $D(f)$ a obor hodnot $H(f)$ spojitý. To znamená, že ke každému bodu z $D(f)$ najdeme libovolně blízký bod. Diskrétní funkce má definiční obor tvořen množinou diskrétních bodů. Definiční obor obrazu je definován jako rovinná oblast R:

$$R = \{(x, y); 1 \leq x \geq x_a\,; 1 \leq y \geq y_b\}, \tag{1.1}$$

kde x je souřadnice horizontální osy, y je souřadnice vertikální osy, x_a a y_b jsou maximální souřadnice v obrazu. Názorný obrázek osových souřadnic je zobrazen na straně 5, Obr. 3.

Matematický popis obrazu je modelován do spojité funkce f dvou nebo tří proměnných a říká se mu obrazová funkce. Je popsán funkcí z:

$$z = f(x, y) \tag{1.2}$$

– pro statický obraz,

$$z = f(x, y, t) \tag{1.3}$$

– pro dynamický obraz,

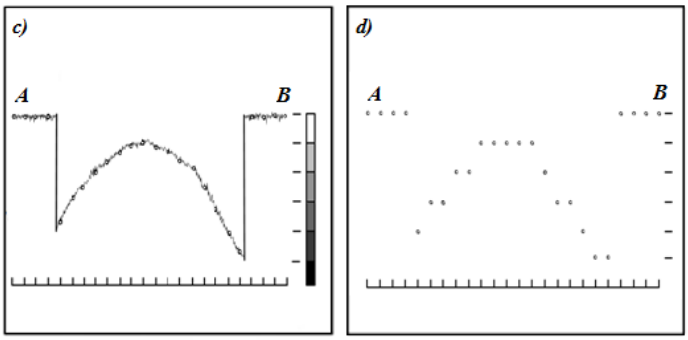
kde x je horizontální souřadnice osy, y je vertikální souřadnice osy, t je třetí proměnná, v případě obrazu měnícího se v čase, f je obrazová funkce dvou nebo tří proměnných a z je hodnota jasu souřadnice x a y. Daný bod odpovídá úrovni jasu v obraze. [4]

Obraz může být monochromatický, tedy reprezentován jedinou obrazovou funkcí $f(x, y)$. Druhou možností je obraz multispektrální (barevný). V tomto případě každé dvojici souřadnic (x, y) odpovídá vektor hodnot, například jasu. Daná práce se zabývá pouze monochromatickým obrazem.

1.1. Digitalizace obrazu

Jestliže chceme zpracovávat obraz na počítači, musíme jej nejprve digitalizovat. Využijeme zde spojitý signál v čase a úrovni, který získáme ze snímání určitého monochromatického obrazu, a převedeme jej do diskrétního tvaru. Toho docílíme přechodem spojité obrazové funkce (1.2) k diskrétní funkci, uvedený proces se nazývá digitalizace. Při zpracování digitálního obrazu se pracuje s digitálními obrazovými funkcemi, které jsou reprezentované maticemi celočíselných hodnot.

Digitalizace obrazu spočívá ve vzorkování obrazu matice dílčích bodů a v kvantování spojité jasové úrovně každého vzorku do jednotlivých intervalů. Čím častěji je prováděno vzorkování a kvantování, tím lépe je aproximován původní obrazový signál.



Obr. 1 Příklad vytvoření digitálního obrazu, c) vzorkování a kvantování, d) digitalizovaná část daného obrazu. [2]

Při digitalizaci obrazu dochází k problémům, mezi které patří nežádoucí vysoké frekvence přispívající k výskytu šumu v obraze. Aliasing vzniká při rekonstrukci daného signálu a je způsoben nedostatečným vzorkováním signálu. Tímto efektem vzniknou nové nízkofrekvenční informace, které se v původním obraze nevyskytovaly. Uvedený stav lze částečně potlačit zvýšením vzorkovací frekvence nebo odstraněním příliš vysokých frekvencí, které nelze vzorkovat. [6]

1.1.1. Vzorkování

Výsledkem vzorkování je diskretizace souřadnic obrazu zjednodušeně se vytvoří matice s $A * B$ body určité velikosti obrazové funkce $f(x, y)$. Jedná se o tzv. prostorové rozlišení. [3]

Pro vzorkování užíváme zpravidla čtvercovou mřížku, která je uvedena na Obr. 4. Interval vzorkování je vzdálenost mezi nejbližšími vzorkovacími body. Interval vzorkování se volí tak, aby byl menší než polovina rozměru nejmenších detailů v obraze.

Vzorkovací frekvence se podle Shannonova teorému volí dvakrát větší, než je maximální požadovaná přenášená frekvence (1.4). Ovšem při zpracovávání obrazu je vhodnější vzorkovat častěji, a to nejméně pětkrát více, než udává Shannonova věta o vzorkování:

$$f_s > 2f_{max}, \tag{1.4}$$

kde f_s je vzorkovací frekvence a f_{max} je maximální frekvence signálu obsažená v obraze.

1.1.2. Kvantování

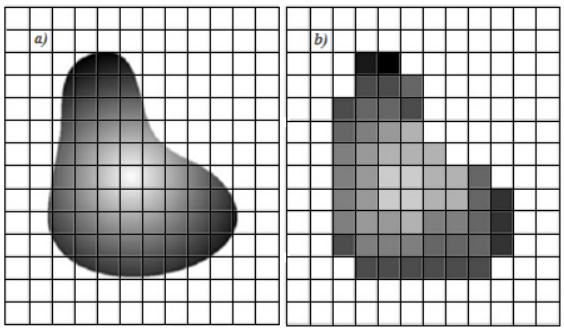
Při kvantování, na rozdíl od vzorkování, dochází k diskretizaci jasových úrovní, tedy k rozdělení spojité jasové úrovně obrazu do k úrovní. Zde se jedná o jasové rozlišení. [3]

Kvantování probíhá v oboru hodnot H obrazové funkce, který je rozdělen na intervaly. Těm je následně přidělena jediná, zástupná hodnota. Množství kvantovacích úrovní musí být dostatečně velké pro vyjádření detailů obrazu. Cílem je přesné vyjádření detailů a zamezení vzniku falešných obrysů v obraze. Kvantování dělíme na uniformní a neuniformní. Uniformní využívá konstantní délku intervalu, pro snadnou realizaci je tato metoda více využívána.

$$k = 2^n, \tag{1.5}$$

kde k je interval udávající počet úrovní jasu a n bitová šířka. Zpravidla se používá osm bitů na jeden obrazový element. Binární obraz je reprezentován informací o obrazovém bodě jedním bitem, logickou jedničku označíme za zájmové objekty a nulu za pozadí. Zmíněné falešné obrysy způsobené nedostatečným kvantováním jsou patrné již při padesáti jasových úrovní.

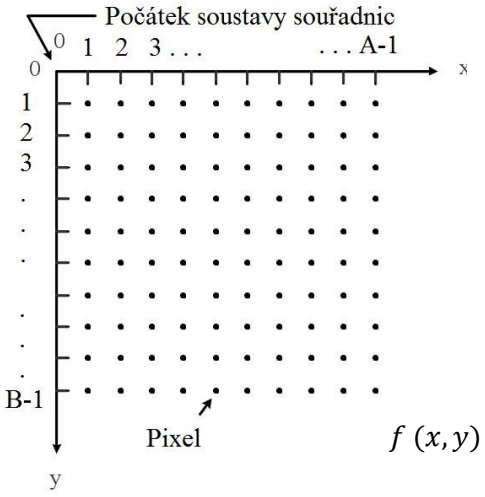
Hodnota kvantovaného signálu se mění skokem, a tak dochází k určité ztrátě informace. To způsobuje takzvané kvantizační chyby, které se projevují především na plochách s malou změnou gradientu, jako náhlý skok barev. Tato chyba působí rušivě a dá se částečně eliminovat neuniformním kvantováním. [6] [7]



Obr. 2 a) Spojitý obraz aplikovaný do vzorkovací mřížky, b) Výsledný obraz vzorkování a kvantování. [2]

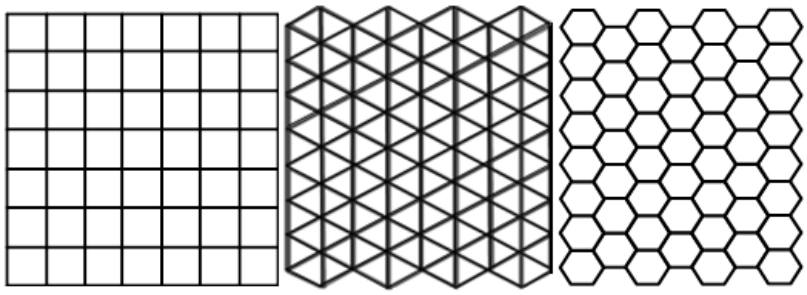
1.2. Reprezentace digitálního obrazu pomocí rastru

Úlohou vnímání obrazu pomocí počítače je naleznout vzájemný vztah mezi vstupním obrazem a modelem z reálného světa. Základní typ zobrazení je rastr neboli bitmapa. Rastr je rozdělen do čtvercové sítě, viz Obr. 4. Ke každému poli této sítě je přiřazena právě jedna hodnota barvy, která odpovídá úrovní jasu v daném bodě. V případě černobílého obrazu hodnota v matici odpovídá: 0 – černá barva, 255 – bílá barva, předpokládáme-li 256 úrovní.



Obr. 3 Reprezentace digitálního obrazu. [2]

Obrazová funkce je představována maticí, jejíž základní a nejmenší jednotkou je obrazový element, takzvaný pixel, viz Obr. 3. Po uspořádání do určitého tvaru rastrové mřížky, viz Obr. 4, pixely pokrývají celý digitalizovaný obraz. Typy jednotlivých rastrových sítí se vyskytují na obrázku níže.



Obr. 4 Příklady tvaru rastrové sítě. Zleva: čtvercová, trojúhelníková a hexagonální síť. [7]

2. SEGMENTACE OBRAZU

Tato kapitola se zabývá nezbytnými postupy při zpracování digitálních snímků. Je zaměřena na segmentaci a zvláště pak na prahování. Výsledky mohou být následně využity například v počítačovém vidění, v našem případě ke zpracování medicínských obrazových dat. Segmentace obrazu je založena zpravidla na detekci hran, které ohraničují objekty nebo na celkové oblasti nacházející se v obraze.

2.1. Základní princip

Při segmentaci obrazu je vstupem intenzitní obraz a výstupem je pak rozčleněný obraz na jednotlivé části. Segmentací se vyjmou z obrazu zájmové objekty a separují se od pozadí, které můžeme chápat jako informační šum. Jelikož dokonalá segmentace není možná, aplikuje se alespoň částečně, kde segmenty obrazu nepředstavují objekty, ale oblasti se stejnou homogenitou. [5]

2.2. Prahování

Segmentace prahováním patří mezi nejstarší a nejjednodušší segmentační metodu. Díky jednoduchosti techniky a snadné implementaci s časovou nenáročností je velmi často používána. Prahování je založeno na rozdílné úrovňové intenzitě mezi objektem a pozadím obrazu. Nadefinuje se práh a každý pixel, který má menší hodnotu intenzity než nadefinovaný práh, je označen jako pixel pozadí, a zbylé pixely spadají do zájmové oblasti objektu, viz vztah (2.1). Výhodou je, že výsledek prahování dostáváme již po jediném průchodu obrazu, a to postupně pixel po pixelu.

Obecně je prahování binárního obrazu definováno následovně. Transformací vstupního obrazu $f(x, y)$ na výstupní získáme binární obraz $g(x, y)$. Určíme jej podle vztahu:

$$g(x,y) = \begin{cases} 1 \rightarrow f(x,y) \geq A \\ 0 \rightarrow f(x,y) < A, \end{cases} \tag{2.1}$$

kde A je předem nadefinovaná konstanta práhu, $f(x, y)$ je obrazová funkce. Určení prahové hodnoty lze určit manuálně i automaticky.

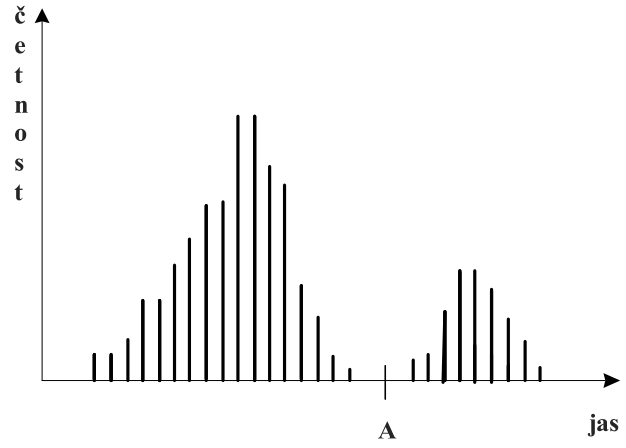
2.2.1. Absolutní prahování

K nejčastěji užívaným metodám můžeme zařadit absolutní prahování, také známé pod názvem globální prahování, kde je dán jednotný práh pro celkový obraz. Uvedený způsob není vyhovující pro složitější obrazy, nicméně pro jednoduché, jakými jsou krevní buňky nebo psaný text jsou naprosto dostačující.

Pro daný případ platí prahování podle vztahu:

$$g(x,y) = \begin{cases} 1 \; pro \; f(x,y) \in A, \\ 0 \; \; jinak, \end{cases} \tag{2.2}$$

kde $g(x,y)$ je binární obraz a A je množina jasové úrovně. Tímto způsobem lze segmentovat obraz na oblasti, jejichž jas náleží množině A a na ostatní oblasti obrazu.



Obr. 5 Segmentace obrazu absolutním prahováním. [2]

Histogram absolutního prahování je vykreslen v závislosti hodnoty četnosti na jasu. Na horizontální ose je hodnota jasové úrovně (0–255) a na vertikální ose je četnost pixelů, které se v daném bodě intenzity vyskytují. Obraz je tvořen ze světlých bodů tvořících objekt a z tmavých bodů náležících pozadí.

Extrahování objektů z pozadí se provádí pomocí prahovací úrovně A, která od sebe oddělí pozadí a zájmové objekty. Zmíněná metoda se provádí dle uvedeného vztahu (2.2).

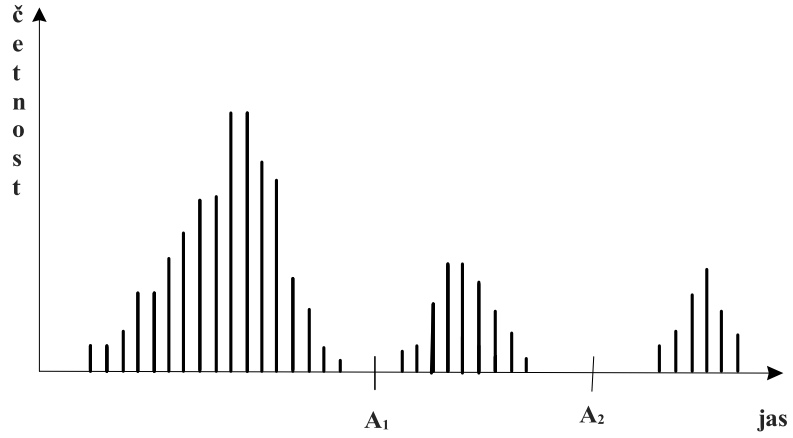
2.2.2. Víceúrovňové prahování

Při absolutním prahování se vyskytují chyby, které jsou způsobeny například odlišným úhlem nasvícení při získávání obrazu. V daném případě lze využít prahování s proměnným prahem, kdy je jeho hodnota určená dle lokálních vlastností daného obrazu.

Takzvané prahování s více práhy, u kterého již není výsledkem binární obraz, ale obraz s omezeným počtem jasových úrovní, je uvedeno v následujícím vztahu:

$$g(x,y) = \begin{cases} 1 & pro\ f(x,y) \in A_1, \\ 2 & pro\ f(x,y) \in A_2, \\ & \vdots \quad \vdots \\ n & pro\ f(i,j) \in A_n, \\ 0 & jinak, \end{cases} \tag{2.3}$$

kde $g(x,y)$ je výsledný binární obraz a A_i jsou podmnožiny jasových úrovní. [4] [8]



Obr. 6 Segmentace obrazu víceúrovňovým prahováním. [2]

Víceúrovňové prahování se provádí obdobným způsobem, jako absolutní prahování s rozdílem, že je nadefinována více než jedna prahová hodnota. Víceúrovňové prahování se provádí dle výše zmíněného vztahu (2.3).

Další částí segmentační techniky je určování a hledání hranic mezi objekty. Danou problematikou se zabývají následující kapitoly.

3. KONVOLUCE

Konvoluce je důležitou matematickou operací nejen pro zpracování obrazu. Jedná se o operaci dvou funkcí, a to zpracovávané funkce a jádra, na jejíž výsledek může být nahlíženo jako na upravenou původní funkci. Konvoluce může být spojitá a diskrétní.

3.1. Spojitá konvoluce

Konvoluce vyjádřena na základě integrálních vztahů a obsahuje nekonečný počet vzorků. Konvoluce dvourozměrných spojitých funkcí f a h je definována integrálem:

$$I(x,y) = f(x,y) * h(x,y) = \int_{a=-\infty}^{\infty} \int_{b=-\infty}^{\infty} f(x-a, y-b)h(a,b)dadb, \qquad (3.1)$$

kde $I(x,y)$ je výstupní obraz, funkce $f(x,y)$ je vstupní obraz a funkce $h(x,y)$ se nazývá konvoluční jádro. Konvoluční jádro je funkce, která udává, jak se obraz pomocí konvoluce mění. Výsledkem konvoluce je funkce daného obrazu o určité hodnotě intenzity pixelu.

3.2. Diskrétní konvoluce

Pro účel práce je nezbytné pracovat s diskrétní konvolucí, která je určená ke zpracování 2D obrazu v počítačové grafice pro takzvaný odhad derivace, tedy zvýraznění hran. Umožní nahradit integrály sumami, a jeho matematický zápis je následující:

$$I(x,y) = f(x,y) * h(x,y) = \sum_{a=-k}^{k} \sum_{b=-k}^{k} f(x-a, y-b)h(a,b), \qquad (3.2)$$

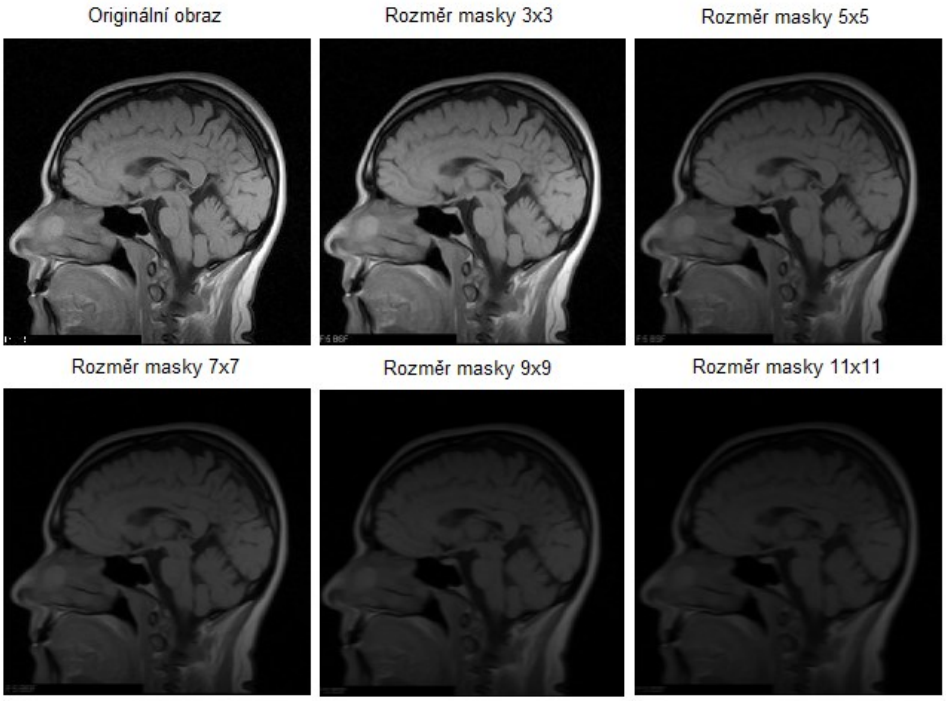
kde $I(x,y)$ je diskrétní obraz, který značí intenzitu výsledného pixelu (x,y), $f(x,y)$ je intenzita vstupního obrazu na pozici (x,y) a $h\,(a,b)$ je konvoluční jádro na pozici (a,b), které také udává její intenzitu. [7]

V případě diskrétní konvoluce si můžeme představit jako konvoluční jádro matici, která se systematicky přikládá k příslušným pixelům v obraze. Názorný příklad a princip implementace konvoluce je popsán v následující kapitole na Obr. 8.

3.2.1 Konvoluční jádra

V počítačovém zpracování obrazových dat využíváme konvoluci obrazu s maskou, která následně aproximuje první nebo druhou derivaci obrazové funkce. Konvoluční maska může být definována různými názvy, jako konvoluční jádro, šablona, matice. U směrových operátorů je tolik masek, kolik směrů daný operátor rozlišuje. Použitá maska pak závisí na požadovaném efektu výsledného obrazu. [4] [9]

Konvoluční maska může mít pouze liché rozměry, tedy 3x3, 5x5, 7x7 pixelů a více. Důvodem je skutečnost, že matice musí mít svůj střed, který je výsledným pixelem v novém obraze po konvoluci. Z hlediska časové náročnosti výpočtu je konvoluce s použitím masky o rozměru 3x3 nejrychlejší. Další výhodou je, že tato maska zanechá původní vlastnosti obrazu ve srovnání s maskami větších rozměrů. Porovnejme, maska 5x5 zanechá více detailů, než o rozměru 9x9 pixelů. Použitím největšího rozměru masky, v našem případě 11x11 pixelů, se obraz nejvíce rozostřil. Můžeme tedy říct, že s použitím větších rozměrů masky se kvalita obrazu zhoršuje. Je rozmazaný, mnohem tmavší a těžko nalézáme jeho detaily.

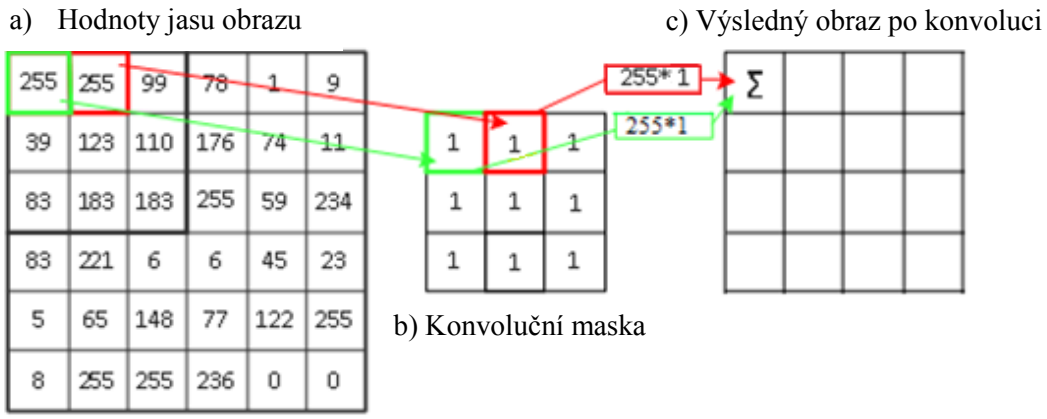


Obr. 7 Ilustrace vlivu rozměru masky na snímek.

Na dané obrázku je zobrazeno aplikování jednotlivých konvolučních masek o různých rozměrech. Použitím větší konvoluční masky se také zvětšuje hranice obrazu. To znamená, že šířka hranice obrazu je rovna poloviční velikosti šablony. Existují okrajové podmínky, které udávají, že výpočet konvoluce lze provézt pouze pro pixely, které jsou dostatečně daleko od okraje daného snímku. Je to způsobeno tím, že nelze k okrajovým pixelům přiložit konvoluční masku tak, aby nepřesahovala hranice původního obrazu. Tomuto efektu se vyhneme rozšířením snímku o okraj pixelů podél celého obrazu (tzv. padding). Daný region lze vyplnit například nulami. Metoda je vhodná v případě předpokladu, že zájmový objekt se vyskytuje ve středu obrazu. Další možností je, že se obraz replikuje do nekonečna podél všech okrajových bodů, tedy rozšíříme okraj po celém obvodu o stejné hodnoty původní hranice snímku. Poslední možností je použít pro rozšířený okraj menší konvoluční jádra. [13]

3.2.2. Implementace konvoluční šablony

Základem je maska, která se aplikuje na příslušné místo v obraze. Je utvořená z pixelů, ve většině případů se užívá matice o rozměru 3x3, popřípadě 5x5. Názorný příklad konvoluce je na Obr. 8. Konvoluci můžeme vypočítat s jedinou maskou, máme-li však více masek, vybere se ta, která dokáže nejlépe aproximovat obrazovou funkci.



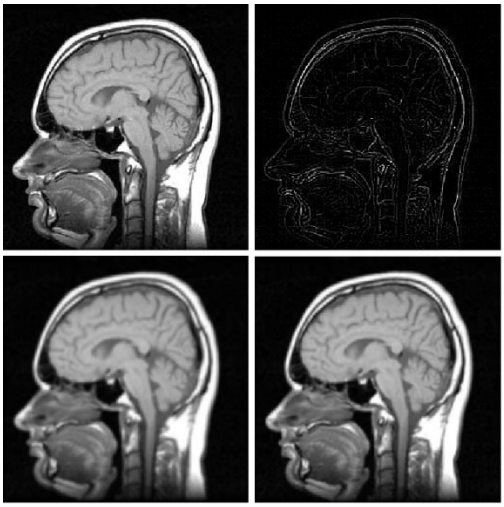
Obr. 8 Princip diskrétní dvourozměrné konvoluce. [11]

Konvoluční maska se postupně přikládá po obraze. Počátkem masky je levý horní roh a z něj se postupuje směrem doprava. Provede se součin každého koeficientu masky s hodnotou pixelu v obraze, který je překryt maskou. Po vynásobení se provede součet všech těchto devíti součinů. Výslednou hodnotu zapíšeme jako nový obrazový bod, který je jakýmsi váženým průměrem okolních pixelů. Po přechodu přes celý řádek se posune konvoluční maska o řádek níže a stejnou cestou postupuje směrem doprava a provádí výpočet až do konce, tedy do pravého dolního rohu.

Mějme příklad výpočtu uvedeného z Obr. 8. Výstupní pixel $f(x, y)$ pak vypočítáme:

$$f(x,y) = 255 * 1 + 255 * 1 + 99 * 1 + 39 * 1 + 123 * 1 + 110 * 1 + 83 * 1 + 183 * 1 + 183 * 1 = 1330 \quad (3.3)$$

Aby nedocházelo ke zvyšování jasu originálního obrazu, můžeme tomuto stavu předejít pomocí předem daných koeficientů, které se vloží před konvoluční masku. Nový výsledný pixel je v podstatě průměrem z devíti okolních pixelů, v případě počítání s maskou o rozměru 3x3. V tomto případě je použit koeficient 1/9. V případě použití masky 5x5 je koeficient roven 1/25 a obraz je více rozostřen. Z uvedeného příkladu vyplývá, že v místě obrazu s konstantním jasem je odezva masky nulová. Konvolučními maskami lze nadefinovat operace, jakými jsou rozostření (Gaussovo zašumění), zaostření (filtrace) či detekci hran (maska aproximující derivaci).



Obr. 9 Ukázka použití konvolučních masek, a) originální snímek, b) s detekovanými hranami, c) zašuměný, d) odšuměný.

Níže jsou zobrazeny konvoluční masky, které byly použity pro uvedený snímek a jejich stručný popis.

$$\text{b) } \mathrm{h}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}, \mathrm{h}_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{3.4}$$

$$\text{c) } h_1 = \frac{1}{159}\begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix} \tag{3.5}$$

$$\text{d) } h_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, h_2 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}, h_3 = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \tag{3.6}$$

Na snímku (b) je použit Laplaceův gradientní operátor, který slouží k detekci hran. Využívá se dvou konvolučních masek (pro čtyř-sousedství a osmi-sousedství). Uvedený operátor aproximuje druhou derivaci. Snímek (c) odpovídá použití Gaussova filtru, který slouží k rozostření obrazu. Můžeme se proto setkat s pojmem Gaussovo zašumění. Poslední snímek (d) odpovídá aplikaci konvolučních masek pro odstranění šumu. Základem je první maska h_1, což je obyčejné průměrování. Někdy se může váha středového pixelu zvyšovat, jak vidíme v maskách h_2 a h_3, ty se pak vytvářejí analogicky. Po sečtení dílčích prvků matice s koeficientem vyjde vždy hodnota jedna.

4. HRANY A HRANOVÉ DETEKTORY

Hranou se rozumí místo v obraze, kde lidský zrak vnímá výraznou změnu jasu. Hrany patří mezi nejzákladnější charakteristiky obrazu. Je nutné si uvědomit, že nevznikají pouze na místech přechodu dvou různých objektů, ale také v případě, kdy dochází k ostré změně barvy v obraze. Odlesky, které jsou způsobeny nesprávným nasvícením, nebo šum přítomný v obraze, mohou zapříčinit výskyt hran. Není vždy lehké určit přesné místo změny mezi dvěma sousedními pixely, proto je vhodné zabývat se prahovou hodnotou podrobněji. Hrany je také možné zvýraznit pomocí kontrastu. Detekce hran je nezbytným klíčem pro segmentaci obrazu.

Pokud potřebujeme obraz zjednodušit, tedy redukovat množství obrazových dat, zaměříme se pouze na nejpodstatnější místa, a to právě na hrany. Zbytek dat můžeme zanedbat a vzniklý obraz pouze z obrysů objektů nezmění myšlenku obsahu obrazu.

4.1. Gradient jasové funkce

Jednotlivé sousedící body obrazu buď na danou hranu navazují, nebo v ní obsaženy nejsou. Ke zjištění návaznosti používáme metody využívající gradient jasové funkce $f(x,y)$. Nástrojem pro změnu funkce dvou proměnných jsou parciální derivace popisující změnu úrovně jasu ve směru osy x a y. Změnu funkce udává gradient. Gradient obrazové funkce je proměnný vektor daný dvěma složkami, velikostí a směrem. Velikost hrany je velikost gradientu (označíme $|\nabla f(x,y)|$) a směr hrany ψ rotuje v závislosti směru gradientu φ. Směr gradientu odpovídá směru největšího nárůstu hodnoty směrem od černé po bílou (od 0 do 255). Směr hrany je ke směru gradientu kolmý, právě zde dochází k nejmenší změně, viz Obr. 10. [7] [10]

Pro obrazovou funkci $f(x,y)$ je velikost gradientu $|\nabla f(x,y)|$ dána vztahem:

$$|\nabla f(x,y)| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}, \tag{4.1}$$

kde $|\nabla f(x,y)|$ je velikost gradientu, tedy velikost hrany v bodě (x,y), $\frac{\partial f}{\partial x}$ a $\frac{\partial f}{\partial y}$ jsou derivace, které popisují změnu úrovně jasu ve směru os x a y. Operátory lze použít k hledání hran, které jsou rovnoběžné se souřadnými osami. Při hledání hran obecného směru je nutné vypočítat průběh jasu ve směru kolmém na směr potenciální hrany. [5]

Směr gradientu φ v bodě (x,y) je dán vztahem:

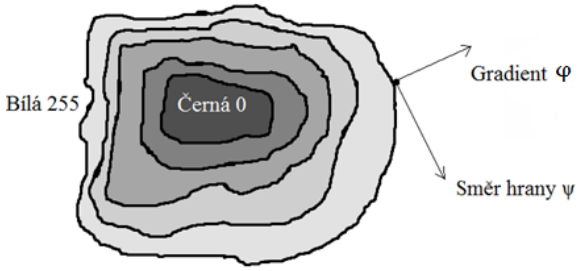
$$\varphi(x,y) = arctg\left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right), \tag{4.2}$$

kde $arctg(x,y)$ je úhel v radiánech mezi souřadnou osou x a radiusvektorem k bodu (x,y).

Směr hrany ψ v bodě (x, y) pak vypočítáme:

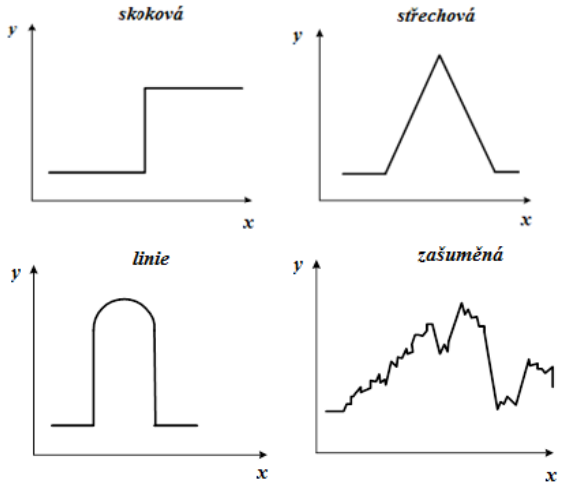
$$\psi(x, y) = \varphi(x, y) + \frac{\pi}{2} \tag{4.3}$$

Vypočtením získáme informaci, zda počítaný bod náleží hranici nějakého objektu či nikoli. V jednoduchém případě můžeme za body hranice považovat místa, kde hodnota velikosti hrany je větší než určitá pevná zvolená prahová hodnota. Uvedený postup má také nedostatek, a to takový, že hranice objektu vyjde větší než jeden bod. [5]



Obr. 10 Směr hrany ψ je kolmý na směr gradientu φ. [7]

Existují čtyři základní modely hran. V reálných obrazech se vyskytuje změna jasu postupně, nikoli skokově. Na obrázku níže jsou znázorněné různé typy jasových profilů hran.



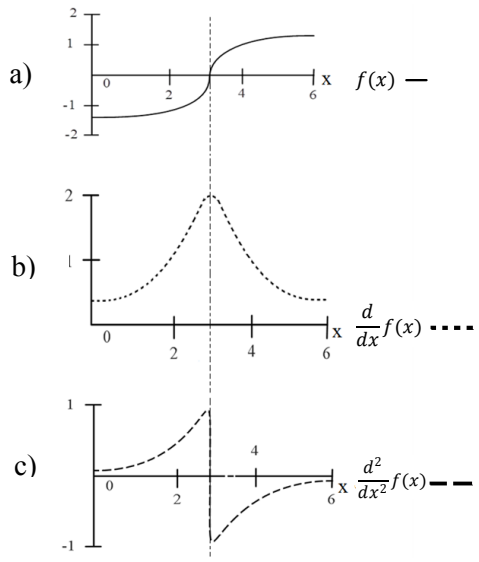
Obr. 11 Jasové profily nejběžnějších hran. [7]

První tři zobrazené profily hran, tedy skoková, střechová hrana a tenká linie, jsou ideální. Zašuměná hrana se vyskytuje v reálném obraze.

4.2. Gradientní operátory

Hranové detektory se liší použitím konvolučního jádra. Lze jimi definovat vyhlazování, ve kterém se používá filtr typu dolní propust, zaostřování, v tomto případě se použije filtr typu horní propust, nebo také detekce hran pomocí gradientních operátorů.

Detekci hran lze rozdělit do tří kategorií gradientních operátorů. Do první skupiny patří operátory, které využívají derivaci prvního řádu pomocí diference. V tomto případě je výsledný gradient porovnán s prahem, který určuje, zda se jedná o hranu či nikoli. Jsou to operátory, které využívají několik konvolučních masek odpovídající určité orientaci (směru). Ta, která nejlépe aproximuje obrazovou funkci, je zvolena jako nejvhodnější pro použití. Příkladem tohoto typu detektoru je Sobelův nebo Prewittové. Laplaceův operátor, který nepodléhá změně vzhledem k rotaci, a může být počítán konvolucí s jedinou konvoluční maskou, spadá pod detektory aproximující druhou derivaci, což je metoda využívající průchody nulou druhé derivace (v angličtině zero-crossing). Je-li změna v obraze dostatečně výrazná, pak je hrana detekována. Třetím typem je aproximace obrazové funkce; princip je založen na základě porovnání jednoduchým parametrickým modelem hran, například polynomem dvou proměnných. [10]



Obr. 12 Průběh jasu a jeho první a druhá derivace v místě hrany. [1]

Obrázek a) odpovídá průběhu obrazové funkce skrz hranu, následně b) je zobrazen průběh této křivky pomocí první derivace a c) je znázorněn průběh křivky pomocí druhé derivace. Je zřejmé, že druhá derivace průběhu jasu ve směru skrz hranu nabývá dvou extrémů, a to kladného a záporného znaménka. V místě, kde se hrana nachází, se znaménko mění. Jelikož směr hrany v obrazu není znám, je nutné počítat druhou derivaci minimálně ve dvou směrech. Při hledání polohy hrany v místě, kde obrazová funkce prochází nulou je díky strmosti přechodu mnohem spolehlivější, než u hledání maxima první derivace.

4.2.1. Detekce hran pomocí první derivace

Dokázali jsme, že se hrana nachází v místě, kde se vyskytuje vysoká hodnota první derivace jasové funkce. Při praktické implementaci počítáme s diskrétní funkcí, tedy nahradíme derivace diferencí. Aproximace derivace je vhodná, chceme-li zachytit změnu signálu, a tím zachytit vyskytující se hrany v obraze. Vlastností první derivace je ta, že dokáže měnit intenzitu úrovně skokově a přitom nedává odpověď na signály, které se nemění. [1]

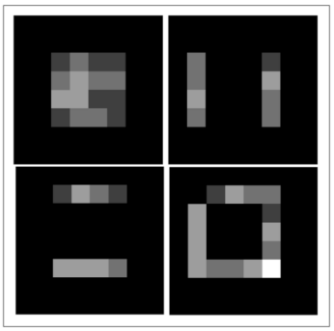
Výpočet diference je relativně jednoduchý, výsledkem je rozdíl sousedních bodů obrazu. Patřičné body se počítají postupně po řádcích a sloupcích. Horizontálního detektor má tvar:

$$|\nabla f_x(x,y)| = |f(x,y) - f(x+1,y)\,| \qquad \forall x \in 1, N-1; y \in 1 \qquad (4.4)$$

Diference počítáme zvlášť pro horizontální a vertikální hranu. Odečtením horizontálních sousedních bodů paradoxně detekujeme vertikální změnu intenzity, tedy nalezneme vertikální hranu. Změny intenzity se ve vodorovném směru nezobrazí, protože je tento rozdíl nulový.

$$\left|\nabla f_y(x,y)\right| = |f(x,y) - f(x,y+1)| \qquad \forall x \in 1, N; y \in 1, N-1 \qquad (4.5)$$

Pro detekci horizontálních hran je právě naopak nutné vertikálního hranového detektoru, viz vztah (4.5), který odečítá vertikální sousední body. Uvedený operátor detekuje horizontální změnu intenzity.



Obr. 13 Detekce hran pomocí první derivace. [1]

Na Obr. 13 jsou zobrazeny dílčí kroky k detekci hran pomocí první derivace. V levém horním rohu je originální obraz zobrazený ve stupních šedi. V pravé horní části je horizontální hranový detektor. V levém dolním rohu je vertikální detektor a poslední obraz odpovídá sloučení těchto detektorů. Na obrázku v levém horním okraji hrany je prázdné místo. Důsledkem je to, že na daném místě ani jeden z detektorů hranu nenalezl. Naopak v pravém dolním rohu je absolutně bílý bod, z důvodu detekce hrany oběma detektory.

Pro náročnost výpočtu velikosti gradientu, viz vztah (4.1), můžeme odmocninu vypustit. Zjednodušený vztah praktický výpočet příliš nezmění.

$$|\nabla f(x,y)| = |\nabla f_x(x,y)| + |\nabla f_y(x,y)| \tag{4.6}$$

Po upravení a dosazení dostaneme:

$$|\nabla f(x,y)| = |f(x,y) - f(x+1,y) + f(x,y) - f(x,y+1)| \tag{4.7}$$

$$|\nabla f(x,y)| = |2 * f(x,y) - f(x+1,y) - f(x,y+1)| \tag{4.8}$$

Daná rovnice odpovídá koeficientům v konvoluční šabloně, viz Obr. 14. Konvoluční maska h, která odpovídá této rovnici má následující tvar:

2	-1
-1	0

Obr. 14 Konvoluční maska $h_{(x,y)}$ pro první derivaci. [1]

Mějme rovnici první derivace:

$$\frac{\partial f(x,y)}{\partial x} = \frac{f(x+\Delta x) - f(x)}{\Delta x} - 0(\Delta x) \tag{4.9}$$

Rozdíl mezi sousedními body udává odhad první derivace s chybou $0(\Delta x)$. Tato chyba závisí na velikosti intervalu Δx a složitosti křivky. Čím je větší interval, tím větší je také chyba. Ta může být redukována tím, že neodečítáme sousední pixely, ale pixely vyskytující se na řádku ob jeden obrazový bod. Nyní je výpočet pro horizontální detektor následující:

$$|\nabla f_{xx}(x,y)| = |\nabla f(x+1,y)| + |\nabla f(x,y)| =$$

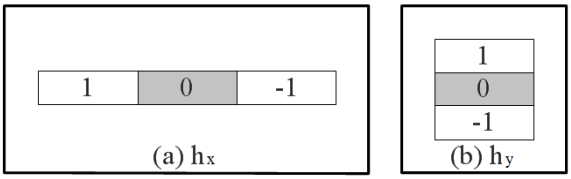
$$= f(x+1,y) - f(x,y) + f(x,y) - f(x-1,y) =$$

$$= f(x+1,y) - f(x-1,y), \tag{4.10}$$

Vertikální hranový detektor pak má následující tvar:

$$|\nabla f_{yy}(x,y)| = f(x,y+1) - f(x,y-1) \tag{4.11}$$

Nyní máme výpočet pro diferenci dvou bodů oddělených jedním pixelem pro směry x a y. Uvedený vztah odpovídá následujícím konvolučním šablonám h_x a h_y.



Obr. 15 Konvoluční masky pro první derivaci s menší chybou. [1]

Na obrázku a) je maska h_x horizontálního detektoru, který detekuje ve svém středovém pixelu vertikální hranu. Na obrázku b) je transponovaná verze tohoto detektoru, maska h_y.

4.2.2 Detekce hran pomocí druhé derivace

Tato metoda je založena na detekci hran v obraze na místě, kde prochází druhá derivace obrazové funkce nulou. V principu je jednodušší hledat místo, kde obrazová funkce protíná osu v bodě nula, viz Obr. 12 c), než lokální extrémy, na kterých je založena detekce hran pomocí aproximace první derivace, viz Obr. 12 b). Výsledkem je hrana, která má konkrétně definované hodnoty, je přesná a označena právě jedním pixelem.

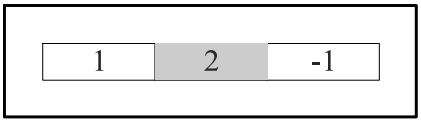
Druhá derivace aproximuje diferencí mezi dvěma sousedními body první derivace:

$$\frac{\partial^2 f(x,y)}{\partial x^2} \cong \frac{\partial f(x,y)}{\partial x} - \frac{\partial f(x,y)}{\partial (x+1)}, \tag{4.12}$$

po upravení dostaneme vztah:

$$\frac{\partial^2 f(x,y)}{\partial x^2} \cong -f(x) + 2f(x+1) - f(x+2), \tag{4.13}$$

a tímto vztahem pak získáme detekci horizontální hrany druhé derivace.



Obr. 16 Horizontální hranový detektor druhé derivace. [1]

Vertikální hranový detektor má stejné hodnoty koeficientů šablony, s rozdílem, že jej zapíšeme do svislého směru. Sloučením horizontálního a vertikálního hranového detektoru získáme Laplaceův operátor, který můžeme považovat za základní konvoluční masku pro detektory aproximující druhou derivaci.

0	-1	0
-1	4	-1
0	-1	0

Obr. 17 Laplaceův hranový detektor.[1]

4.3 Přehled hranových detektorů

Pro zjednodušení a přehlednější zápis hranových detektorů aproximující první derivaci ve směru osy x a y, zavedeme označení hodnot obrazové funkce A, B, C, D, F, G, H, I. Označené hodnoty jsou pro oblast obrazu o rozměru 3x3 pixelů. Jejich uspořádání je následující:

A	B	C
D	*f(x,y)*	F
G	H	I

Obr. 18 Značení obrazových hodnot pro určitou oblast obrazu. [5]

4.3.1. Robertsův hranový detektor

Robertsův detektor považujeme za jeden z nejstarších detektorů vůbec, jeho realizace výpočtu velikosti hrany je nejjednodušší. Byl navržen Lawrencem Robertsem roku 1936. Využívá matici pouze o rozměru 2x2 sousedních pixelů. Nejčastěji se využívá pro obrazy ve stupních šedi. Je vytvořen ze dvou masek, které jsou zrcadlově otočeny.

Velikost hrany (gradientu) v bodě (x, y) se vypočítá:

$$|\nabla f(x,y)| = |f(x+1, y+1) - f(x,y)| + |f(x, y+1) - f(x+1, y)| \qquad (4.14)$$

Použijeme-li nově zavedené značení obrazových hodnot, pak je vztah následující:

$$|\nabla f_x\,(x,y)| = |f(x,y) - I| \qquad (4.15)$$

$$\left|\nabla f_y\,(x,y)\right| = |F - H| \qquad (4.16)$$

Konvoluční maska má tvar:

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}, \tag{4.17}$$

kde h_1 odpovídá výpočtu velikosti hrany $|\nabla f_x\,(x,y)|$ a h_2 odpovídá výpočtu hrany $|\nabla f_y\,(x,y)|$.

Podle velikosti konvoluční masky je možné odvodit, že nevýhodou detektoru je velká citlivost na šum. Je to z důvodu malého okolí použitého pro odhad gradientu. V neprospěch hovoří také slabá reakce na skutečné a velmi ostré hrany. Aplikace masky pro horizontální a vertikální hranu je pro detekci hran v obraze samostatná, následným spojením těchto masek se najde absolutní velikost gradientu v každém bodě orientace. [10]

Důvodem navržení těchto konvolučních masek je rychlá a jednoduchá implementace, dávající velkou odezvu na hrany. Robertsův hranový detektor nejlépe detekuje hrany, které se vyskytují pod úhlem 45 stupňů a 135 stupňů v pixelové mřížce. [1] [2]

Obr. 19 Detekce hran Roberstovým operátorem (prahování nastaveno na 0,1).

4.3.2. Prewittové hranový detektor

Operátor aproximující první derivaci pomocí diference je hranový detektor Judith Prewittové, který vznikl roku 1966. Jedná se o operátor, který využívá k detekci hran okolí pixelů o velikosti 3x3, a to pro osm směrů. Základem jsou dvě konvoluční masky (h_1, h_2) pro 0 a 45 stupňů, ostatní pak vzniknou pootočením těchto masek. Velikost gradientu je dán maskou udávající maximální odpověď hodnoty gradientu. Konvoluční šablona může mít také větší rozměry, například 5x5, viz (4.21). Operátor Prewittové reaguje nejen na změnu intenzity jasu, ale také na šum, který se vyskytuje v každém obraze. K detekci hran je vhodné začlenit prahování pro odstranění vyskytujícího se šumu v daném obraze. Pro zavedené hodnoty obrazové funkce má výsledný vztah tvar:

$$|\nabla f_x\,(x,y)| = \frac{1}{3}[(C-A)+(F-D)+(I-G)] \tag{4.18}$$

$$\left|\nabla f_y\,(x,y)\right| = \frac{1}{3}[(A-G)+(B-H)+(C-I)], \tag{4.19}$$

kde $|\nabla f_x\,(x,y)|$ je výsledná hodnota velikosti hrany ve směru x a $\left|\nabla f_y\,(x,y)\right|$ je výsledná hodnota hrany ve směru y. Detektor Prewittové je založen na výpočtu aproximace derivace ve směrech os x a y průměrem Ve vzorcích (4.18) a (4.19) se vyskytují tři součtové operace, z tohoto důvodu je hodnota koeficientu $\frac{1}{3}$. V případě, kdy nezáleží na velikosti absolutní hodnoty hrany, můžeme uvedené koeficienty vypustit. Jestliže následuje po výpočtu velikosti hrany prahování, můžeme koeficienty nahradit volbou velikosti prahu. [5] [7]

Nyní je zobrazeno čtyři kombinace šablon pro realizaci operátoru Prewittové:

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}, h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, h_4 = \begin{bmatrix} -1 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix} \tag{4.20}$$

Veškeré konvoluční šablony Prewittové jsou citlivé na orientaci hrany od nuly do 315 stupňů v krocích po 45 stupních, přičemž nula odpovídá svislé hraně. Příklad vícerozměrné masky operátoru Prewittové, konkrétně pro detekci svislých hran:

$$\begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix} \tag{4.21}$$

Obr. 20 Detekce hran pomocí operátoru Prewittové.

4.3.3. Sobelův hranový detektor

Daný detektor byl prezentován roku 1968 Irwinem Sobelem. Je velmi podobný operátoru Prewittové. Operátor je opět sestaven z masky pro vodorovný a svislý směr o rozměru matice 3x3 pixelů. [5]

Pro výpočet derivací v jednotlivých směrech používá Sobelův operátor vážený průměr:

$$|\nabla f_x\,(x,y)| = \frac{1}{4}[(C-A)+2(F-D)+(I-G)] \tag{4.22}$$

$$\left|\nabla f_y\,(x,y)\right| = \frac{1}{4}[(A-G)+2(B-H)+(C-I)] \tag{4.23}$$

Výpočet hrany je opět ve směru os x a y a je zde použito zavedených obrazových hodnot. I v tomto případě lze koeficientů $\frac{1}{4}$ vypustit.

Konvoluční masky pro aplikaci Sobelova operátoru:

$$h_1 = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix},\quad h_2 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix},\quad h_4 = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & 2 \end{bmatrix}, \tag{4.24}$$

kde h_1 aproximuje derivaci ve směru x a h_2 aproximuje derivaci ve směru y. Sobelův operátor se používá především pro detekci vodorovných a svislých hran, proto nám postačí pouze masky h_1 a h_2. Pro zajímavost se vyskytuje i maska h_3, která je pro úhel 45 stupňů a maska h_4 pro úhel 135 stupňů.

Obr. 21 Detekce hran pomocí Sobelova operátoru.

4.3.4. Kirschův hranový detektor

Kirschův hranový detektor je podobný operátorům uvedených výše, byl popsán Russell Kirschem roku 1971. I nyní můžeme daný hranový detektor zařadit mezi operátory založené na výpočtu gradientu pomocí aproximace první derivace. Kirschův operátor je v praktické části použit pro detekci cévního řečiště oka a očního nervu.

Pro stručnost si opět zavedeme náhradní parametry obrazové funkce, viz Obr. 22 Pro lepší přehled zavedeme hodnoty $A_0, A_1, \dots, A_7$.

A_0	A_1	A_2
A_7	$f(x,y)$	A_3
A_6	A_5	A_4

Obr. 22 značení hodnot obrazové funkce pro Kirschův operátor. [5]

Výpočet velikosti hrany v bodě (x, y) se provádí podle následujícího vzorce:

$$|\nabla f_x\,(x,y)| = max_{i=1}^{7}[|5 * S - 3 * T_i|], \tag{4.25}$$

kde

$$S_i = A_i + A_{i+1} + A_{i+2}, \tag{4.26}$$

$$T_i = A_{i+3} + A_{i+4} + A_{i+5} + A_{i+6} + A_{i+7}, \tag{4.27}$$

přičemž index i, pro který je ve vztahu (4.25) dosaženo maxima, určuje směr dané hrany. V případě, že v uvedených vztazích (4.26) nebo (4.27) vyjdou hodnoty indexů větší než 7, upraví se modul operací o osmičku. [5]

Konvoluční masky Kirschova detektoru obsahují prvky 0, 3 a -5. Ostatní čtyři masky vzniknou pouhým pootočením.

$$h_1 = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix} h_2 = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}$$

$$h_3 = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix} h_4 = \begin{bmatrix} -5 & -5 & 3 \\ -5 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix} \tag{4.28}$$

Obr. 23 Detekce hran Kirschovým operátorem.

4.3.5. Laplaceův hranový detektor

Laplaceův hranový detektor patří mezi gradientní operátory, které aproximují druhou derivaci. Jeho vlastností je, že udává stejnou odezvu ve všech směrech, je tedy invariantní vzhledem k natočení souřadné soustavy. Proto je vhodné využit Laplaciánu pouze v případě, jestliže nám postačí detekovat jen hrany, nikoliv směr. Výpočetní nenáročnost je dána právě invariancí rotace detekce. Jeho kladnou vlastností je, že je vhodný pro ostření obrazu. Nevýhody Laplaceova detektoru zahrnují příliš velkou citlivost na šum a dvojité odezvy. [7]

Obecný vzorec pro výpočet velikosti hrany $|\nabla f(x,y)|$ pomocí druhé derivace vypadá následovně:

$$|\nabla^2 f(x,y)| = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2} \tag{4.29}$$

kde ∇^2 je označení pro všesměrový Laplaceův operátor.

Je-li obrazová funkce diskrétní, použijeme namísto derivací diference. Výsledný předpis pro realizaci Laplaceova operátoru je pak následující:

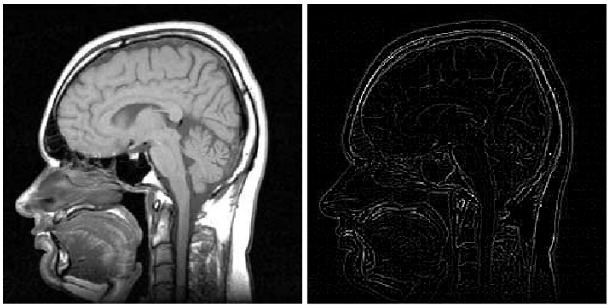
$$|\nabla^2 f(x,y)| = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{4.30}$$

Dvě základní používané masky pro čtyři (h_1) a osm sousedství (h_2) se vyskytují níže.[10]

$$h_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} h_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.31}$$

Můžeme se setkat s případem aplikace Laplaciánu, který využívá větší váhu pixelů blízko reprezentativního bodu masky. V tomto případě se ztratí invariantnost vzhledem k otočení. [10]

$$h_1\begin{bmatrix} 2 & -1 & 2 \\ -1 & -4 & -1 \\ 2 & -1 & 2 \end{bmatrix}, h_2\begin{bmatrix} -1 & 2 & -1 \\ 2 & -4 & 2 \\ -1 & 2 & -1 \end{bmatrix} \tag{4.32}$$



Obr. 24 Detekce hran Laplaceovým operátorem.

4.3.6. Marrův-Hildrethové filtr

Zhotoviteli uvedeného filtru byli v sedmdesátých letech neurolog David Marr a profesorka počítačové vědy Ellen Hildrethová. U operátorů aproximujících druhou derivaci je značná citlivost na šum a proto je potřeba zkombinovat výpočet konvoluce s vyhlazovacím filtrem. Vyhlazení musí být dostatečně výrazné pro odstranění šumu, avšak ne natolik, aby porušilo hrany vyskytující se v obraze. Takto popsaný Marrův-Hildrethové operátor vznikne složením Laplaciánu a Gaussiánu. [1] [2]

Filtr tohoto typu by měl splňovat určité požadavky. Měl by být hladký a pohybovat se v pásmové propusti frekvenčního spektra z důvodu omezení počtu frekvencí, kde může dojít k průchodu nulou. Mezi další požadavek můžeme zařadit přesnost lokalizace hrany, to znamená, že filtr reaguje jen na body, které se vyskytují v blízkosti okolí hrany. Jelikož jsou výše zmíněné požadavky v protikladu, je potřeba využít lineárního filtru, který odpovídá 2D Gaussovu rozložení $G(x, y)$:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.33}$$

kde x, y jsou souřadnice obrazu a σ je směrodatná odchylka, která udává, v jak velkém okolí operátor pracuje. Například pixely vyskytující se blíže středu mají větší váhu. [10]

4.3.7. Operátor Laplacián Gaussiánu

Detektor hran využívající Laplacián obrazové funkce po vyhlazení Gaussovým filtrem, za použití konvoluce, je znám pod názvem „Laplacián Gaussiánu“, se zkratkou LoG ($\nabla^2 G$).

Po vysvětlení a uvedení Laplaciánu (∇^2), Gaussiánu (G) a s použitím druhé derivace můžeme napsat vztah pro LoG operátor:

$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} = \frac{\partial}{\partial x}\left[\frac{-x}{\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}}\right] + \frac{\partial}{\partial y}\left[\frac{-y}{\sigma^2}\, e^{-\frac{x^2+y^2}{2\sigma^2}}\right] =$$

$$= \left[\frac{x^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} + \left[\frac{y^2}{\sigma^4} - \frac{1}{\sigma^2}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}, \tag{4.34}$$

po úpravě vztahu (4.34) dostaneme výsledný vztah pro LoG operátor:

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.35}$$

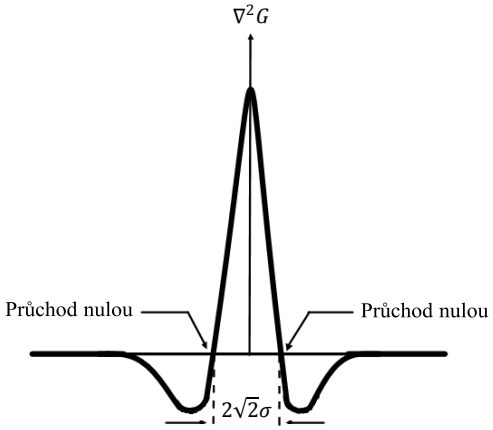
Rovnice konvoluce pro Laplacián Gaussiánův operátor má pak následný tvar:

$$LoG = [\nabla^2 G(x,y,\sigma)] * f(x,y) = \nabla^2 [G(x,y,\sigma) * f(x,y)], \tag{4.36}$$

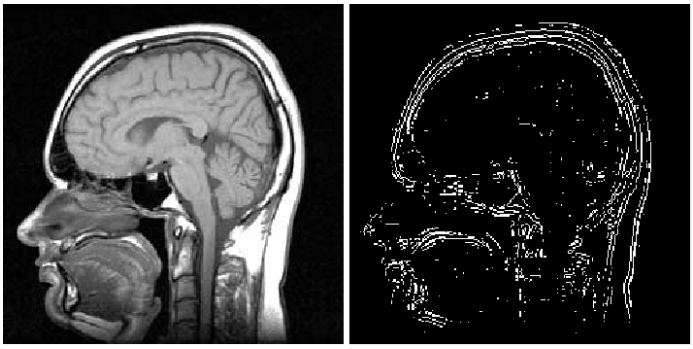
kde σ je směrodatná odchylka. Vzhledem k linearitě lze pořadí operací konvoluce prohodit. Aproximace operátoru LoG konvoluční masky h_1 o rozměru 5x5 vypadá následovně:

$$h_1 = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix} \tag{4.37}$$

Díky svému tvaru je maska také nazývána „mexický klobouk“.



Obr. 25 Příčný řez operátoru LoG s vyznačením průchodů nulou. [2]



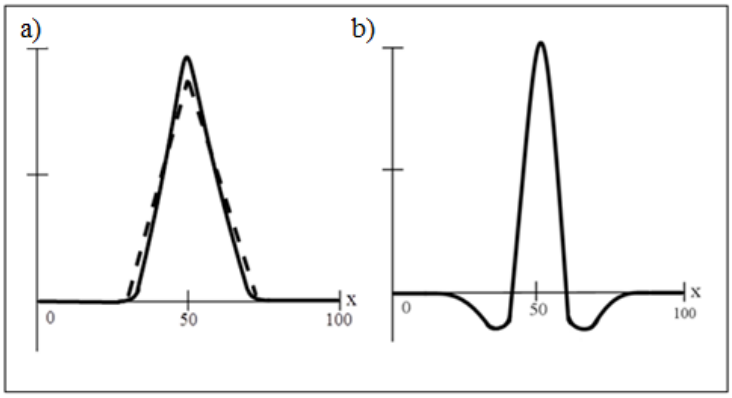
Obr. 26 Operátor Laplacián Gaussiánu.

4.3.8. DoG Operátor

Existuje druhý způsob využití Marrův-Hildrethové operátoru, a to rychlé a efektivní aproximace s použitím diference dvou obrazů o různé směrodatné odchylce. Nazveme jej pak "diference Gaussiánů", zkratkou je označován jako DoG. Vzorec pro výpočet diference Gaussiánů:

$$DoG(x,y) = \frac{1}{2\pi\sigma_1^2} e^{-\frac{x^2+y^2}{2\sigma_1^2}} - \frac{1}{2\pi\sigma_2^2} e^{-\frac{x^2+y^2}{2\sigma_2^2}}, \tag{4.38}$$

přičemž $\sigma_1 > \sigma_2$, což je směrodatná odchylka. Příklad aplikace DoG operátoru může být následující:



Obr. 27 Aproximace LoG operátoru s diferencí Gaussiánů. [1]

a) $e^{-\left(\frac{x-50}{8}\right)^2}$ ▬, 0,8 $e^{-\left(\frac{x-50}{10}\right)^2}$ ▬ ▬ b) $e^{-\left(\frac{x-50}{8}\right)^2} - 0{,}8e^{-\left(\frac{x-50}{10}\right)^2}$ ▬

Mezi nevýhody operátorů LoG a DoG patří příliš velké vyhlazení obrazu, ztráta ostrosti rohů a při nesprávném nastavení hodnoty práhu mají operátory sklony utvářet uzavřené obrysy, a tím znehodnotit vlastnosti obrazu. [9] [10]

4.3.9. Cannyho hranový detektor

Uvedený detektor byl publikován v roce 1986 Johnem F. Cannym a mohli bychom jej považovat za nejlepší a teoreticky za ideální hranový detektor. Základní myšlenkou Cannyho detektoru je představa, že skokovou hranu (ve 2D obrazu si ji představme jako schod) můžeme najít pomocí filtru. Cannyho detektor je založen stejně jako Laplaceův detektor na hledání průchodů nulové hodnoty. Cannyho detektor zahrnuje několik kroků k dosažení co nejlepších výsledků při detekci hran. Canny navrhnul tři základní požadavky, které by měl detektor splňovat. Požadavky jsou následující:

a) minimalizovat pravděpodobnost chybné detekce. To znamená, že detektor nemá detekovat žádné falešné (neexistující) hrany, ale nalézt všechny podstatné.

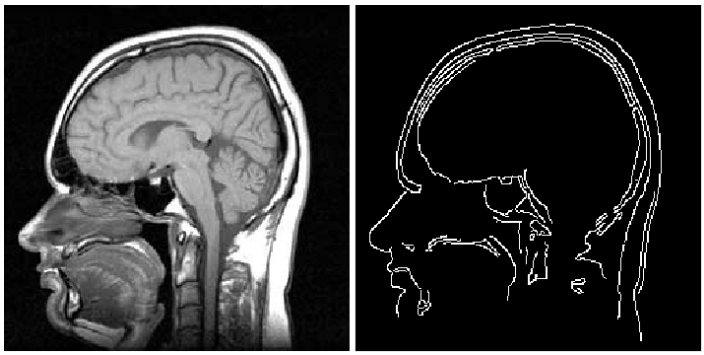
b) najít co nejpřesněji polohu dané hrany v obraze. Tímto je myšleno nalézt správnou lokalizaci hrany, aby vzdálenost mezi skutečnou a detekovanou hranou byla co nejmenší. [10]

c) pouze jedna odezva. Každá hrana by měla být detekována pouze jedenkrát.

Detekce hran pomocí Cannyho detektoru je následující. Prvním krokem je eliminace šumu, kterou zajistíme Gaussovým filtrem, ale je možné použít i jiné filtry. Poté nalezneme směr gradientu a derivaci ve směru pro každý pixel pomocí vhodné konvoluční masky.

Dalším nezbytným krokem při hledání hran je prahování, kterým nalezneme lokální maxima zmíněných derivací. Jedná se o prahování s hysterezí, to znamená, že si nastavíme maximální a minimální hodnotu prahu.

Jedním z problémů při detekci hran Cannyho detektorem je určení správné hodnoty směrodatné odchylky (σ) v případě použití Gaussova filtru. Při zvolení malé hodnoty σ detektor nalezne i malé a bezvýznamné hrany. Naopak při větší hodnotě σ slabé hrany zaniknou a zhorší se lokalizace také ostatních hran.



Obr. 28 Cannyho hranový detektor (prahování – 0,5, σ – 0,1).

5. DETEKCE HRAN V PROSTŘEDÍ MATLAB

Cílem práce je segmentace biomedicínských snímků na základě konvoluce a detektorů hran, které jsou realizovány v prostředí programu MATLAB (verze 2011a). V první části jsou rozebrány dostupné funkce, se kterými MATLAB disponuje, následně je práce zaměřena na návrh a implementaci vlastních funkcí pro detekci hran.

5.1. Dostupné hranové detektory funkce edge v MATLABu

MATLAB pro segmentaci obrazu na základě detektorů hran disponuje s určitými hranovými detektory, které si nyní podrobněji rozebereme spolu s jejich možnosti nastavení parametrů. Základní funkcí pro detekování hran v MATLABu je funkce *edge*. Tato funkce nalézá hrany obrazu ve stupních šedi. Její syntaxe funkce je následující:

OBR = edge (I),

kde *I* je název proměnné původního binárního obrazu nebo obrazu ve stupních šedi pro funkci *edge*. Výsledek této funkce pak vrací binární obraz *OBR* (název proměnné výstupního obrazu) o stejném rozměru, jako je původní vstupní obraz *I,* kde logická hodnota jedna odpovídá místu v obraze. V případě, že hrana detekována nebyla, přiřadí tomuto bodu hodnotu logické nuly.

5.1.1 Sobelův hranový detektor

Nenastavíme-li určitý typ hranového detektoru, se kterým MALAB disponuje, výchozím nastavením je Sobelův detektor. Sobelova metoda je založena na hledání hran pomocí aproximace první derivace. Hrany výsledného snímku odpovídají bodům, jejichž výsledná hodnota gradientu je maximální. Nyní si podrobně rozebereme jednotlivé možnosti nastavení pro detekci hran pomocí Sobelova operátoru. Od vzorového kódu se liší zápisem, nyní specifikován typ hranového operátoru, se kterým se pracuje. Výsledek syntaxe, která je definována obecně, je totožný s Obr. 29.

Obr. 29 Zleva: Původní snímek, Sobelův detektor.

Syntaxe kódu v MATLABu vypadá následovně: *BW = edge (I, 'sobel');*

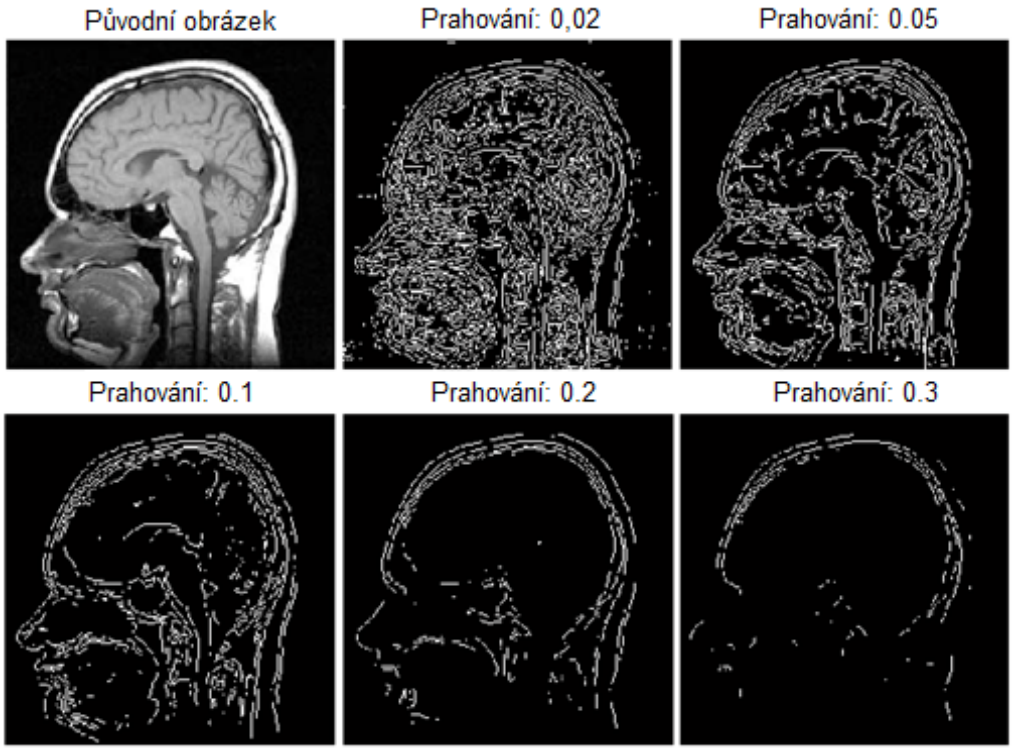
Vzorový kód: *sob1 = edge (obr1, 'sobel');*

Ze syntaxe vyplývá, že se jedná o funkci *sobel*. V případě, že není nastavena žádná hodnota prahování, práh citlivosti je pak vždy zvolen automaticky, a to u všech detektorů.

Syntaxe kódu v MATLABu: *BW = edge (I,'sobel', thresh);*

U Sobelova detektoru můžeme nastavit prahování, jehož hodnota se pohybuje v rozmezí od nuly do jedné, odvozeno z angličtiny od slova tresholding, což znamená prahování. Funkce *edge* ignoruje veškeré hrany, které nejsou silnější, než je nastavená hodnota prahování.

Vliv prahování na určitý snímek po použití Sobelova detektoru je zobrazen níže. Jsou zde použity různé hodnoty citlivosti práhu od nejmenší, což odpovídá hodnotě 0,02 až po největší hodnotu práhu, která je v našem případě 0,3.



Obr. 30 Aplikace Sobelova detektoru s různou citlivostí práhu.

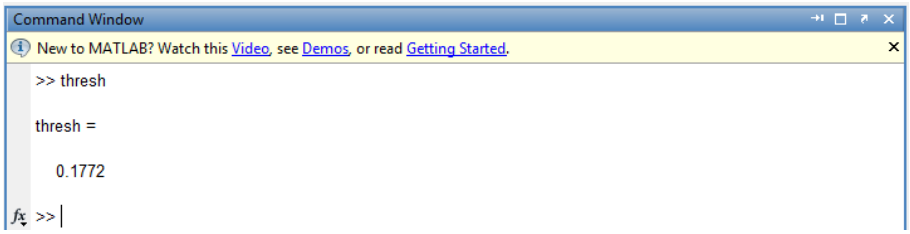
Vzorový kód: *sob = edge (obr1, 'sobel', 0.02); sob2 = edge (obr1, 'sobel', 0.05); sob3 = edge (obr1, 'sobel', 0.1); sob4 = edge (obr1, 'sobel', 0.2); sob5 = edge (obr1, 'sobel', 0.3);*

Nastavení vyšší hodnoty prahování je téměř zbytečné z důvodu ztráty informací z obrazu. Hodnota 0,3 je v případě Sobelova detektoru hraniční pro zjištění, co se na snímku vyskytuje. Z medicínského hlediska je uvedený snímek bezvýznamný a nepoužitelný. Opakem je velmi nízký práh citlivosti, který odpovídá hodnotě 0,02, na němž je opět nelehké rozeznat podstatné části snímku a připomíná spíše shluk bílých čar a teček. Dále se v něm vyskytují artefakty, které odpovídají bílým bodům vyskytující se okolo hlavy. Jestliže je zájmovou oblastí např. část páteře, vhodnou hodnotou prahování je 0,05. Nicméně, za optimální výstup považujeme citlivost prahu kolem hodnoty 0,1.

Syntaxe kódu v MATLABu: [*BW, thresh] = edge (I, 'sobel',...);*

Vzorový kód: *[sob, thresh] = edge (obr1, 'sobel');*

V této fázi se pouze odkážeme na zájmovou hodnotu, do příkazového okna tedy napíšeme slovo *thresh*. Výsledkem pak je hodnota prahování použitá na zvolený medicínský snímek. Pro daný snímek je funkcí *edge* vyhodnocena za vhodnou citlivost prahu hodnota 0,1772.



Obr. 31 Ukázka příkazového okna s výstupní hodnotou prahování Sobelova detektoru.

Syntaxe kódu v MATLABu: *BW = edge (I, 'sobel', thresh, direction);*

Direction, z angličtiny směr, určuje směr detekce hran Sobelova detektoru. *Direction* je řetězec, který udává, zda se detekují hrany horizontální, vertikální nebo obojí. Ve výchozím nastavení je detekce horizontálních i vertikálních hran. Pro nastavení horizontální či vertikální detekce zapíšeme do kódu *'horizontal'* nebo *'vertical'*.

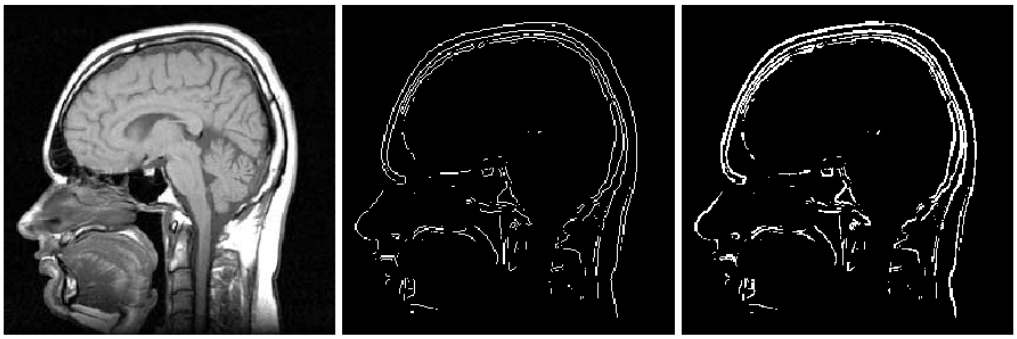
Obr. 32 Zleva: Původní snímek, detekce horizontálních hran, detekce vertikálních hran.

Vzorový kód: *sob2 = edge (obr1, 'sobel', 0.2, 'horizontal'); sob3 = edge (obr1, 'sobel', 0.2, 'vertical');*

Povšimněme si, jak výrazně se výstupní obrazy liší. Prostřední snímek jednoznačně odpovídá detekci horizontálních hran a vertikální hrany se zde téměř nevyskytují. Naopak detekce vertikálních hran, která se vyskytuje na krajním snímku, nalézá pouze svislé hrany.

Syntaxe kódu v MATLABu*: BW = edge (I, 'sobel',..., options);*

Tato funkce umožňuje vložit volitelný vstupní řetězec. Řetězec *'nothinning'* urychluje činnost algoritmu přeskočením přídavné fáze zužování hran. Ve výchozím nastavení, nebo v případě, že je řetězec *'thinning'* specifikován, algoritmus zužování hran aplikuje.



Obr. 33 Zleva: Původní snímek, zúžené hrany, detekce bez fáze zúžení hran.

Vzorový kód: *tic sob4 = edge (obr1, 'sobel', 0.2, 'thinning'); toc sob5 = edge(obr1, 'sobel', 0.2, 'nothinning');*

Funkce zužování hran má za úkol zjemnit hrany a také zajistit a minimalizovat výskyt zdvojených hran. Výpočetní čas detekce s použitím funkce zužování hran je téměř desetinásobná (263 ms) při porovnání s aplikováním řetězce bez zúžení hran (27 ms). Výpočetní časy jsou změřeny pomocí funkce *tic* a *toc.*

Syntaxe v MATLABu: *[BW, thresh, gv, gh] = edge (I, 'sobel',...);*

Daná syntaxe vrací horizontální a vertikální hrany. Konkrétně pak jejich výslednou hodnotu Sobelova gradientu v daném bodě, výsledkem jsou dvě matice, zvlášť pro každý směr.

Vzorový kód: *[sobel,thresh,gv,gh] = edge(obr1,'sobel'),*

kde *gv* odpovídá matici pro detekci vertikálních hran a *gh* odpovídá detekci horizontálních hran.

5.1.2 Detektor Prewittové

Nyní si rozeberme detektor Prewittové a jeho možnosti nastavení. Nastavení se téměř shoduje s možnostmi nastavení u Sobelova detektoru. Rozdílem je především použitá odlišná maska. Pro ukázku je použit jiný snímek zobrazený v sagitální rovině, ale liší se stupněm jasu.

Syntaxe kódu v MATLABu*: BW = edge(I,'prewitt');*

Zmíněným příkazem pouze specifikujeme hranový detektor, který je použit na určitý snímek. V tomto případě nenastavujeme prahování, ani detekci hran v horizontálním či vertikálním směru, dané parametry jsou nastaveny automaticky funkcí *edge.*

Obr. 34 Zleva: Původní snímek, detektor Prewittové.

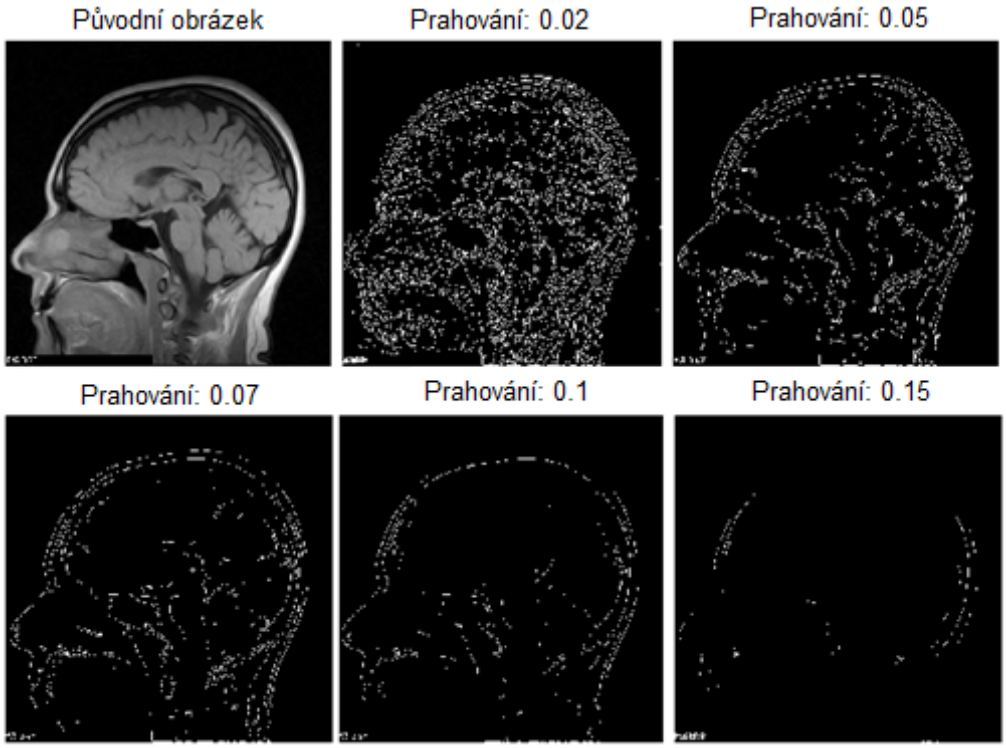
Vzorový kód: *pre1 = edge (obr1, 'prewitt');*

Použitý medicínský snímek je tvořen z mnohem tmavších odstínů, než snímek u Sobelova operátoru. Výsledný snímek je zašedlý a hrany jsou nevýrazné a těžko definovatelné. Pro vylepšení detekce je možné nafotit nový snímek, což je mnohdy komplikované. Také můžeme vyzkoušet jiný detektor, nebo obraz dále zpracovat například zvýrazněním hran.

Syntaxe kódu v MATLABu*: BW = edge (I, 'prewitt', thresh);*

Funkcí *thresh* nastavíme citlivost práhu pro detektor Prewittové. Funkce *edge* ignoruje veškeré hrany, které nejsou silnější, než je nastavená hodnota prahování.

Vzorový kód: *Prewitt = edge (obr1, 'prewitt', 0.02); prewitt2 = edge (obr1, 'prewitt', 0.05); prewitt3 = edge (obr1, 'prewitt', 0.07); prewitt4 = edge (obr1, 'prewitt', 0.1); prewitt5 = edge (obr1, 'prewitt', 0.15);*

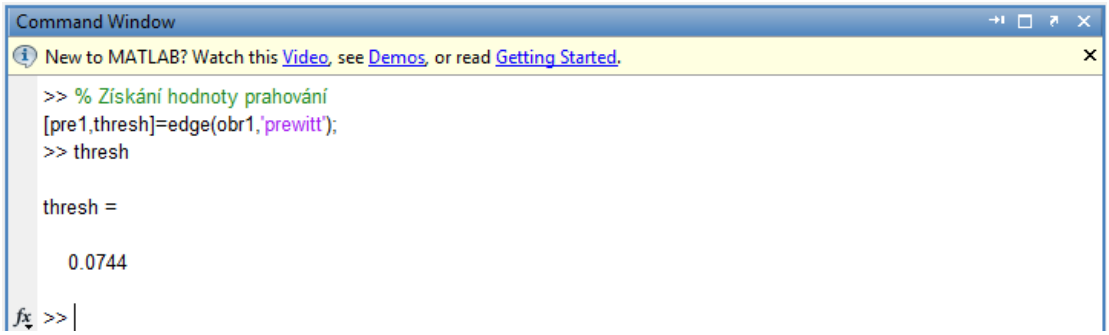


Obr. 35 Aplikace detektoru Prewittové s různou citlivostí práhu.

Z obrázku je patrné, že rozsah prahování je mnohem menší a postačí velmi nízká hodnota k tomu, aby byl výsledný snímek nerozeznatelný. U Sobelova detektoru a použitém světlejším snímku, je hraniční hodnota 0,3 a výsledný snímek je přesto viditelnější než v tomto případě, kde je hodnota prahování nastavená na 0,15. Takto zvolená hodnota prahování je pro další získání informací z obrazu nepřijatelná. Po aplikaci druhého nejvýše zvoleného práhu citlivosti (0,1)výsledný obraz opět ztrácí vlastnosti obrazu, a je tedy nevhodné zvolit danou hodnotu. Hodnota prahování 0,02 je také nevyhovující, z výsledného snímku můžeme získat pouze obrys hlavy a shluk bílých teček. Za optimální nastavení prahování můžeme pro zvolený snímek považovat hodnotu okolo 0,07. Všechny snímky po aplikaci operátoru Prewittové mají stejnou vlastnost, a to nespojité hrany. Výsledné hrany jsou zobrazeny pomocí přerušovaných čar nebo jednotlivých bílých bodů.

Syntaxe kódu v MATLABu: [*BW, thresh] = edge (I, 'prewitt',...);*

Hranový detektor Prewittové disponuje se stejnou funkcí jako Sobelův operátor, a tou je získání hodnoty prahování, která je automaticky nastavena funkcí *edge* v případě, že tuto hodnotu uživatel nenastavil. Pro náš vstupní medicínský snímek je za optimální prahování považována hodnota 0,0744. Podívejme se, jak můžeme pomocí příkazového okna a funkce *thresh* citlivost práhu získat.



Obr. 36 Ukázka příkazového okna s výstupní hodnotou prahování detektoru Prewittové.

Vzorový kód: *[pre, thresh] = edge (obr1, 'prewitt');*

Jestliže použijeme detektor Prewittové na snímek, který je tvořen světlejšími odstíny, například využijeme-li snímek použitý pro Sobelův detektor, výsledná hodnota prahování je 0,1736. Můžeme tedy říct, že nastavení citlivosti práhu pro detekci hran je individuální, podle tónového rozsahu zvoleného původního snímku. Snímek je více či méně zašedlý, nevýrazný, může být tvořen spíše světlejšími nebo tmavšími obrazovými body.

5.1.3 Robertsův hranový detektor

Dalším z hranových detektorů, který nabízí program MATLAB pomocí funkce *edge,* je Robertsův operátor. Ze všech hranových detektorů je jeho výstupní snímek vykreslen nejrychleji. Kód vykazuje nejmenší výpočetní náročnost, důvodem je velikost masky Robertsova operátoru, jejíž rozměr je pouze 2x2 pixelů.

Syntaxe v MATLABu: *BW = edge(I,'roberts');*

Danou syntaxí pouze specifikujeme, jaký operátor je funkcí *edge* použit pro detekci hran pro zvolený snímek. Ostatní možnosti nastavení nejsou upřesněny, funkce *edge* použije své defaultní hodnoty.

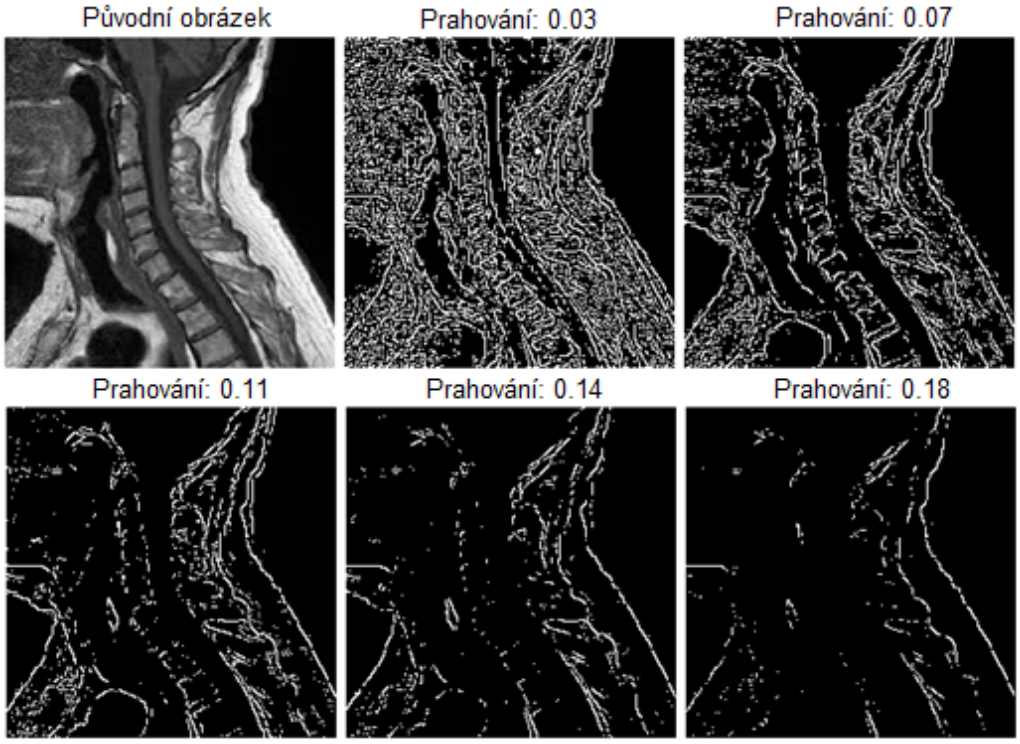
Obr. 37 Zleva: Původní snímek, Robertsův detektor.

Vzorový kód: *roberts1 = edge (obr1, 'roberts');*

Pro ukázku Robertsova detektoru je pro zpestření zvolen jiný medicínský snímek, který je zobrazen v sagitální rovině a je zaměřen především na krční páteř. Z obrázku je patrné, že je nastavena nesprávná hodnota prahování. Ne vždy je výchozí nastavení citlivosti práhu ideální. Můžeme říct, že není vhodné spoléhat pouze na automatické nastavení pomocí funkce *edge*. Je nutné si vyzkoušet různé hodnoty práhu pro jednotlivé medicínské snímky a dále se hlouběji zaměřit na zájmové části daného snímku.

Syntaxe v MATLABu: *BW = edge (I, 'roberts', thresh);*

Nyní si nastavíme hodnotu prahování manuálně. Pro ukázku byly zvoleny pro Robertsův detektor následující hodnoty citlivosti práhu.



Obr. 38 Aplikace Robertsova hranového detektoru s různou citlivostí práhu.

Vzorový kód: *rob = edge (obr1, 'roberts', 0.03); rob2 = edge (obr1, 'roberts', 0.07); rob3 = edge (obr1, 'roberts', 0.11); rob4 = edge (obr1, 'roberts', 0.14); rob5 = edge (obr1, 'roberts', 0.18);*

Přestože je hodnota práhu 0,18 nízká, výsledný snímek už ztrácí podstatné hrany. Páteř a její ohraničení nelze v tomto případě vůbec rozeznat. Prahování nastavené na hodnotu 0,14 nám poskytuje nástin krční páteře, nicméně je přesto daný snímek téměř bezvýznamný.

Jedinou výhodou tohoto snímku je obrys objektu, který se nachází uprostřed snímku, a je ze všech výsledných snímků nejvýraznější. Příliš nízká hodnota prahování je 0,03; obsažený šum zhoršil kvalitu snímku a není na něm možné přesně definovat hrany. Za optimálně zvolenou hodnotu práhu v daném případě považujeme hodnotu 0,07. Výsledek nám poskytuje nejpřehlednější zobrazení jednotlivých částí snímku. Dostatečně lze také rozeznat krční obratle a část obratlů hrudních, mozkomíšní mok a také prodlouženou míchu procházející páteří.

Syntaxe v MATLABu: *BW = edge (I, 'roberts',..., options),*

kde místo *options* použijeme zmíněnou funkci zúžení hran (*'thinning')* nebo *funkci* bez zúžení (*'nothinning'*). Pokud možnost nespecifikujeme, je defaultně použit algoritmus na ředění hran.

Po použití funkce zúžení hran na daný snímek jsou hrany užší a zobrazí se přesněji na místech, kde se vyskytují v původním snímku. To znamená, že výstupní snímek odpovídá vstupnímu. Tímto je vylepšena kvalita a snímek není znehodnocen artefakty, např. zdvojenými hranami. Výpočetní čas je pro danou funkci roven 40 ms, zatímco druhá funkce, při které se fáze zúžení přeskočí, trvá 6,2 ms. Mezi těmito výpočetními časy je více než šestinásobný rozdíl.

Obr. 39 Zleva: Původní snímek, zúžené hrany, detekce bez fáze zúžení hran.

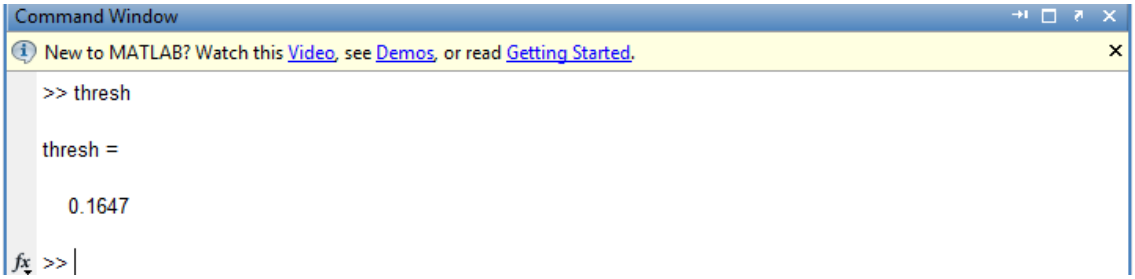
Vzorový kód: *tic roberts4 = edge (obr1, 'roberts', 0.07, 'thinning'); toc roberts5 = edge (obr1, 'roberts', 0.07, 'nothinning');*

Výsledné snímky se liší svou výrazností. V případě použití funkce zúžení hran vypadá snímek nejasný a jeví jako horší. Nicméně pokud se podíváme na snímky ve větším rozměru, výsledný snímek, na který je aplikována fáze zúžení, je lepší a přesnější.

Syntaxe kódu v MATLABu: *[BW, thresh] = edge (I, 'Roberts',...);*

Výsledkem dané syntaxe je opět hodnota prahování, která je automaticky nastavena funkcí *edge* na daný snímek.

Vzorový kód: *[rob, thresh] = edge (obr1, 'roberts');*



Obr. 40 Ukázka příkazového okna s výstupní hodnotou prahování Robertsova detektoru.

Syntaxe kódu v MATLABu: *[BW, thresh, g45, g135] = edge (I, 'roberts',...);*

Vzorový kód: *[rob, thresh, g45, g135] = edge (obr1, 'roberts');*

V případě zadání dané syntaxe do příkazového jsou výstupem matice o velikosti původního snímku. Jednotlivé body v matici odpovídají výsledným hodnotám gradientu Robertsova operátoru. Jelikož Roberstův detektor nalézá pouze šikmé hrany, první matice je pro detekci hran pod úhlem 45 stupňů a druhá pod úhlem 135 stupňů.

5.1.4 Detektor LoG

Lapaceův hranový detektor využívá metodu založenou na detekci hran pomocí druhé derivace. V obraze tedy hledáme místa, kde hodnota druhé derivace prochází nulou.

Syntaxe kódu v MATLABu: *BW = edge (I,'log')*

Danou syntaxí specifikujeme, o jaký detektor se jedná. Nespecifikujeme hodnotu práhu ani σ. V případě, že hodnota σ není nastavena uživatelem, je defaultně nastavena hodnota 2.

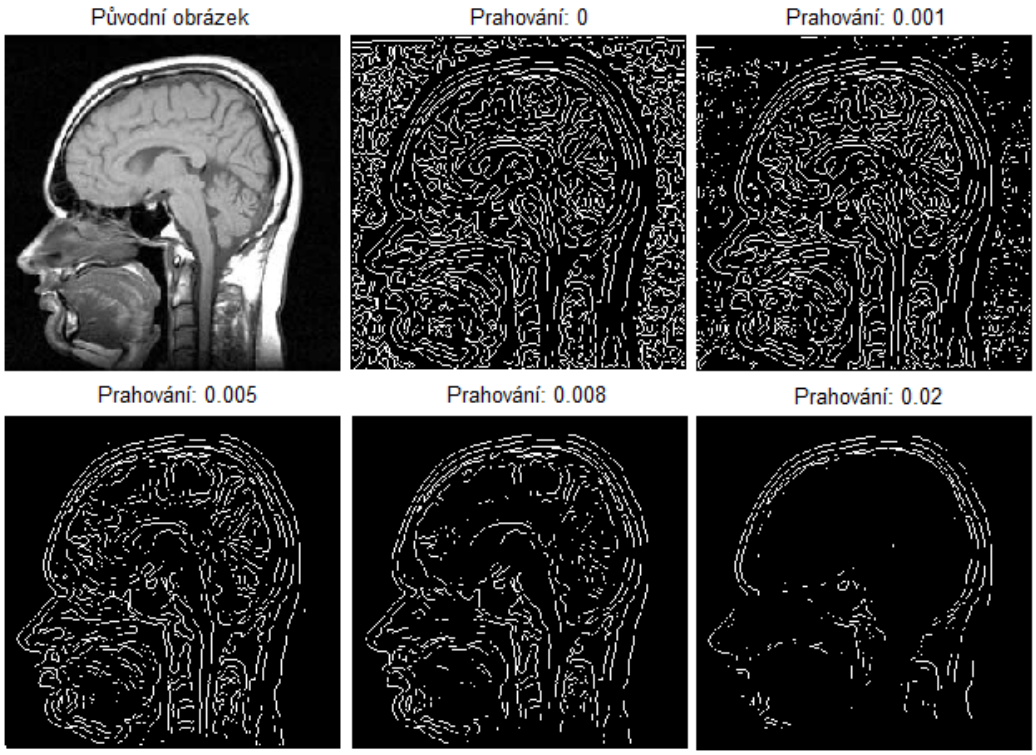
Obr. 41 Zleva: Původní snímek, detektor LoG.

Vzorový kód: *laplacian = edge (obr1, 'log');*

Co se týče přehlednosti a zobrazeným hranám odpovídajícím původnímu snímku, je ze všech zmíněných hranových detektorů nejlepší operátor LoG.

Syntaxe kódu z MATLABu: *BW = edge (I, 'log', thresh);*

Podívejme se, jak se mění výstupní snímek v závislosti na změně hodnoty citlivosti prahu. Ta je zvolena pouze v rozmezí od 0 po 0,02. Nastavovat vyšší hodnotu již nemá význam.

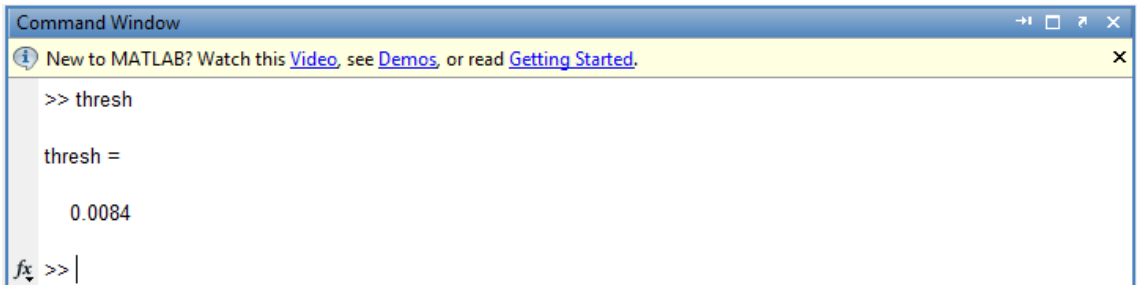


Obr. 42 Aplikace LoG detektoru s různou citlivostí prahu.

Vzorový kód: *log = edge (obr1, 'log', 0); log2 = edge (obr1, 'log', 0.001); log3 = edge (obr1, 'log', 0.005); log4 = edge (obr1, 'log', 0.008); log5 = edge (obr1, 'log', 0.02);*

Po nastavení citlivosti prahu na nulovou hodnotu obsahuje výstupní snímek uzavřené obrysy; je to způsobeno tím, že hodnota zahrnuje všechny průchody nulou vstupního snímku. Přestože je na snímky použit rozsah pouze do hodnoty 0,02, vidíme značný rozdíl mezi dílčími výstupními snímky. Po použití hodnoty prahu 0,001 se nežádoucí šum vyskytuje méně, nicméně je přesto viditelný. Nastavení citlivosti prahu na hodnotu 0,02 není optimální, výsledný snímek neposkytuje žádné informace o hranách vyskytujících se v hlavě. Za vhodně zvolenou citlivost považujeme hodnotu prahu 0,008, která je následně použita i pro různé možnosti nastavení σ.

Syntaxe kódu v MATLABu: *[BW, thresh] = edge (I, 'log');*

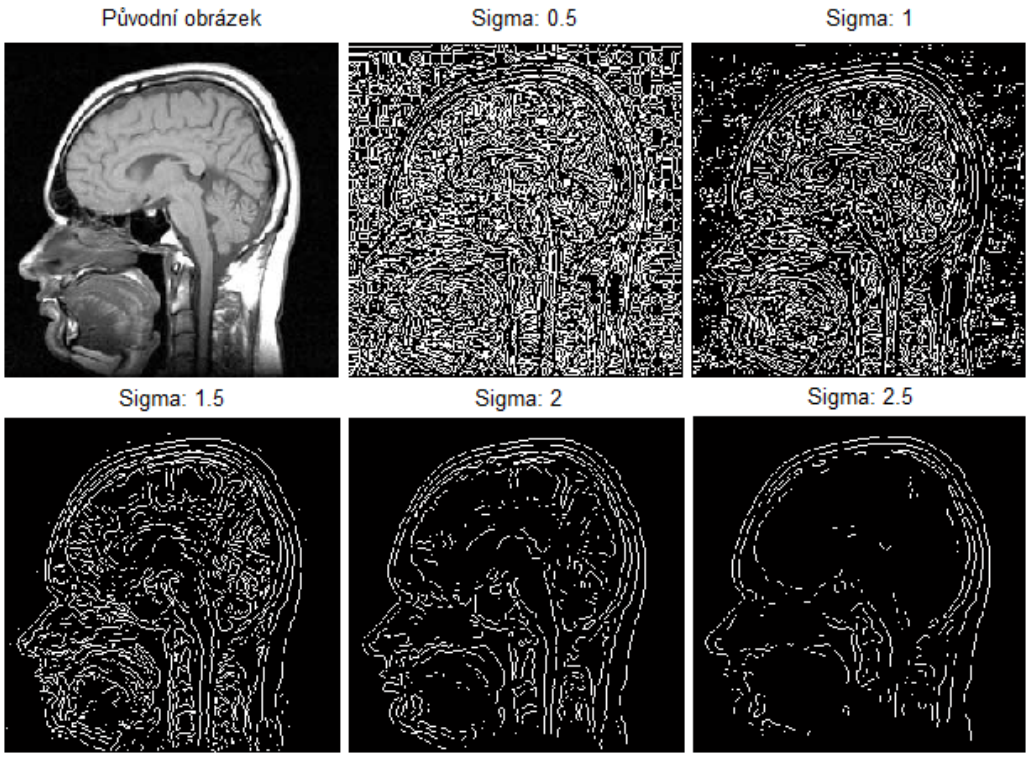


Obr. 43 Ukázka příkazového okna s výstupní hodnotou prahování Robertsova detektoru.

Vzorový kód: *[log, thresh] = edge (obr1, 'log');*

V případě zvoleného snímku a LoG detektoru je za optimální práh považována hodnota 0,0084. Nenastavíme-li hodnotu σ, je nastavena funkcí *edge* automaticky.

Syntaxe kódu v MATLABu: *BW = edge (I, 'log', thresh, sigma);*



Obr. 44 Aplikace LoG detektoru s citlivostí práhu 0,008 a různou hodnotou σ.

Vzorový kód: *laplac = edge (obr1, 'log', 0.008, 0.5); laplac2 = edge (obr1, 'log', 0.008,1); laplac3 = edge (obr1, 'log', 0.008, 1.5); laplac4 = edge (obr1, 'log', 0.008, 2); laplac5 = edge (obr1, 'log', 0.008, 2.5);*

Pro ukázku nastavení σ, je zvolena pevná hodnota prahování (0,008). První dvě nastavené hodnoty 0,5 a 1 nejsou zvoleny správně, na výsledném snímku je obsažen šum a objekty, které absolutně neodpovídají původnímu snímku. Výsledný snímek při nastavené σ na 1,5 obsahuje šum, avšak v minimálním množství. Všimněme si nežádoucích hran obsažených ve snímku, které jsou přítěží. Po nastavení σ na hodnotu 2,5 se šum nevyskytuje, nicméně z výsledného snímku získáme pouze informaci, kde se nachází obrys hlavy. Nevýhodou jsou také vyskytující se přerušované hrany, které lemují tvar lebky. Za optimálně nastavenou σ v daném případě považujeme hodnotu 2. Můžeme říct, že s rostoucí hodnotou σ dojde k většímu potlačení šumu a také zanikne větší množství slabých hran. Nevýhodou je pak menší přesnost lokalizace hran a dochází k jejich dvojitým odezvám. Při nesprávně nastavené σ a prahování se ve výstupním snímku zobrazí spojené hrany tvořící obrazce, které nejsou shodné s původním snímkem. Další nevýhodou je, že se po aplikaci LoG detektoru zobrazují hrany a různé objekty, jež se v původním snímku ani nevyskytují (černé pozadí kolem hlavy).

5.1.5 Zero-crossing

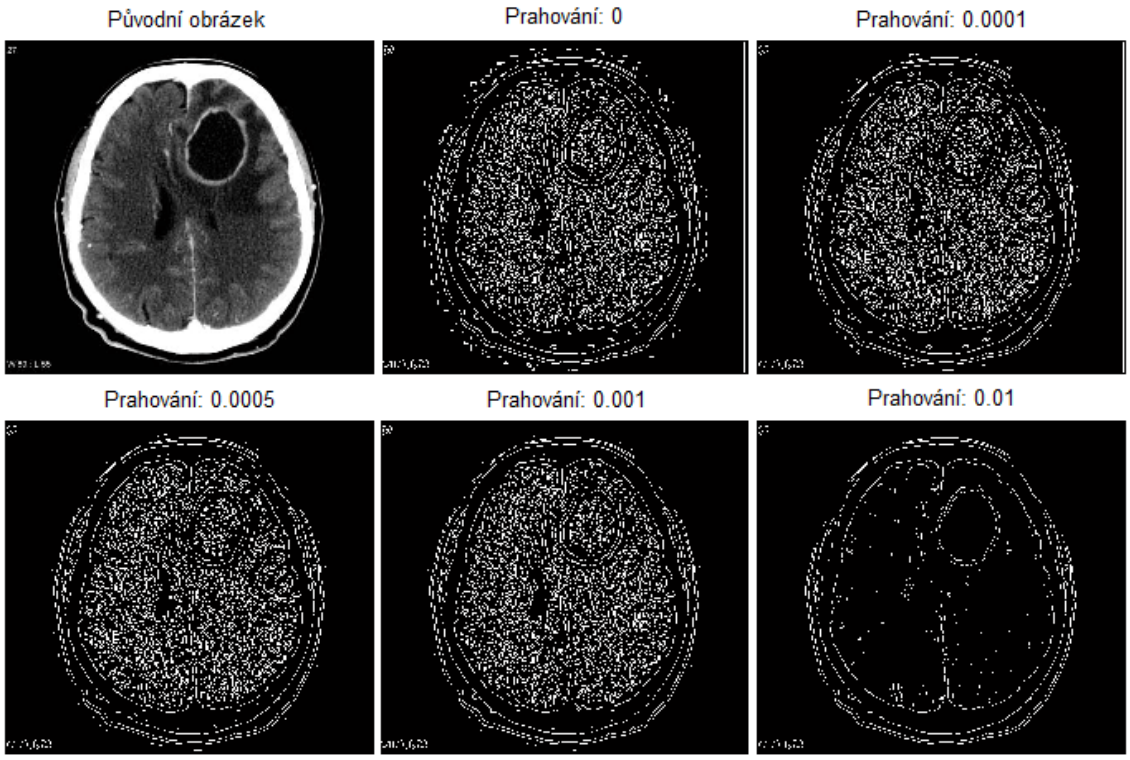
Zmíněná detekce hran je založena na průchodu druhé derivace obrazové funkce nulou, známá pod názvem zero-crossing.

Syntaxe kódu v MATLABu: *BW = edge (I, 'zerocross', thresh, h);*

Daná syntaxe specifikuje konkrétní metodu detekce hran. Hodnota citlivosti práhu je opět nastavitelná. Jestliže zadáme nulovou hodnotu práhu, výsledný snímek utvoří uzavřené obrysy, stejně jako u LoG operátoru. Použitím filtru *h* specifikujeme metodu zero-crossing.

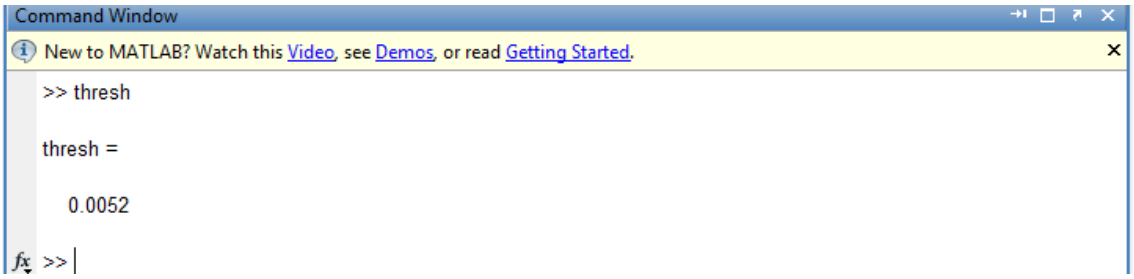
Pro ukázku detekce hran pomocí metody průchodu druhé derivace nulou je zvolen medicínský snímek mozku z počítačové tomografie, který je zobrazen v axiální rovině po aplikaci kontrastní látky. Na Obr. 45 je zjevný rozsáhlý patologický útvar nacházející se v pravé horní části mozku. Pro specifikaci filtru *h* je využit Laplaceův filtr. Po nastavení práhu na nulovou hodnotu se sice nevyskytuje velké množství šumu, nicméně přesto je výsledný snímek nepřehledný, kolem okraje hlavy je příliš nežádoucích hran. Z výsledných snímků můžeme využít snímky vyskytující se ve spodní části Obr. 45. Jestliže chceme přesně detekovat ohraničení mozku, vhodným nastavením práhu je hodnota mezi 0,0005 – 0,001. V případě detekce patologického útvaru pak zvolíme vyšší hodnotu práhu, a to 0,01. Útvar je po použití daného práhu ohraničen a přehledně lze také pozorovat rozdělení mozku na hemisféry

Vzorový kód: *zero1 = edge (obr1, 'zerocross', 0, 'log'); zero2 = edge (obr1, 'zerocross', 0.0001, 'log'); zero3 = edge (obr1, 'zerocross', 0.0005, 'log'); zero4 = edge (obr1, 'zerocross', 0.001, 'log'); zero5 = edge (obr1, 'zerocross', 0.01, 'log');*



Obr. 45 Aplikace metody zero-crossing s různým nastavením citlivosti práhu.

Syntaxe kódu v MATLABu: *[BW, thresh] = edge (I, 'zerocross',...);*



Obr. 46 Příkazové okno s výstupní hodnotou prahování při metodě zero-crossing.

Vzorový kód: *[zerocross, thresh] = edge (obr1, 'zerocross');*

Výsledkem zmíněné syntaxe je nastavená hodnota *funkcí* edge pro zvolený snímek mozku. V daném případě je za optimální citlivost práhu považována hodnota 0,0052.

5.1.6 Cannyho hranový detektor

Posledním z dostupných detektorů funkce *edge* je Cannyho detektor. Ze všech detektorů je nejlepší právě zmíněná metoda. Cannyho detektor se od ostatních detektorů liší prahováním.

Používá dvě různé hodnoty práhu, a to pro možnost nastavení detekování silnějších a slabších hran v obraze. Další výhodou je detekování slabších hran pouze v případě, že jsou spojené se silnějšími. Tímto zabráníme výskytu nežádoucího šumu a existuje větší pravděpodobnost, že nalezneme skutečné slabé hrany namísto nepotřebného šumu.

Syntaxe kódu v MATLABu: *BW = edge (I, 'canny');*

Obr. 47 Zleva: Původní snímek, Cannyho detektor.

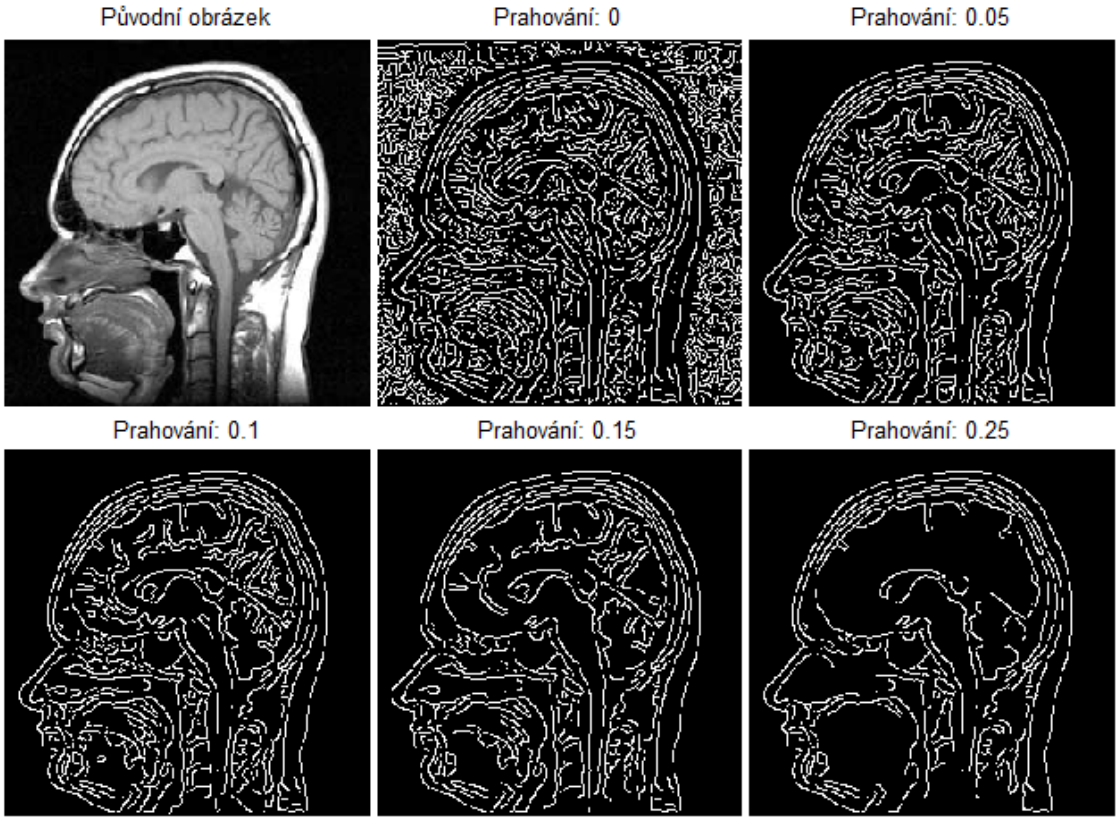
Vzorový kód: *canny = edge (obr1, 'canny');*

Z výstupního snímku je zřejmé, že Cannyho detektor je nejlepší ze všech zmíněných detektorů. Výstupní snímek informuje uživatele o veškerých podstatných hranách a obrysech vyskytujících se v obraze. Zřetelně jsou zobrazeny mozkové závity a jejich oddělení pomocí zářezů. Hranice mezi páteří a míchou je také dostatečně rozeznatelná, a veškeré zobrazené hrany jsou nepřerušované. Na výsledném snímku se nevyskytuje žádný šum i přesto, že není specifikována citlivost práhu ani hodnota σ.

Syntaxe kódu v MATLABu: *BW = edge (I, 'canny', thresh);*

Nastavuje se vymezená hranice, při které se hrany zobrazí. Pokud nastavíme jen jednu hodnotu práhu, Cannyho detektor ji považuje za nejvyšší práh. Nejnižší práh je pak vypočten jako násobek koeficientu 0,4 se zadaným prahem. Nezadáme-li žádnou hodnotu citlivosti práhu, funkce *edge* opět automaticky zvolí optimální hodnotu vzhledem k nejvyšší hodnotě velikosti gradientu daného snímku.

Vzorový kód: *Canny = edge (obr1, 'canny', 0); canny2 = edge (obr1, 'canny', 0.05); canny3 = edge (obr1, 'canny', 0.1); canny4 = edge (obr1, 'canny', 0.15); canny5 = edge (obr1, 'canny', 0.25);*

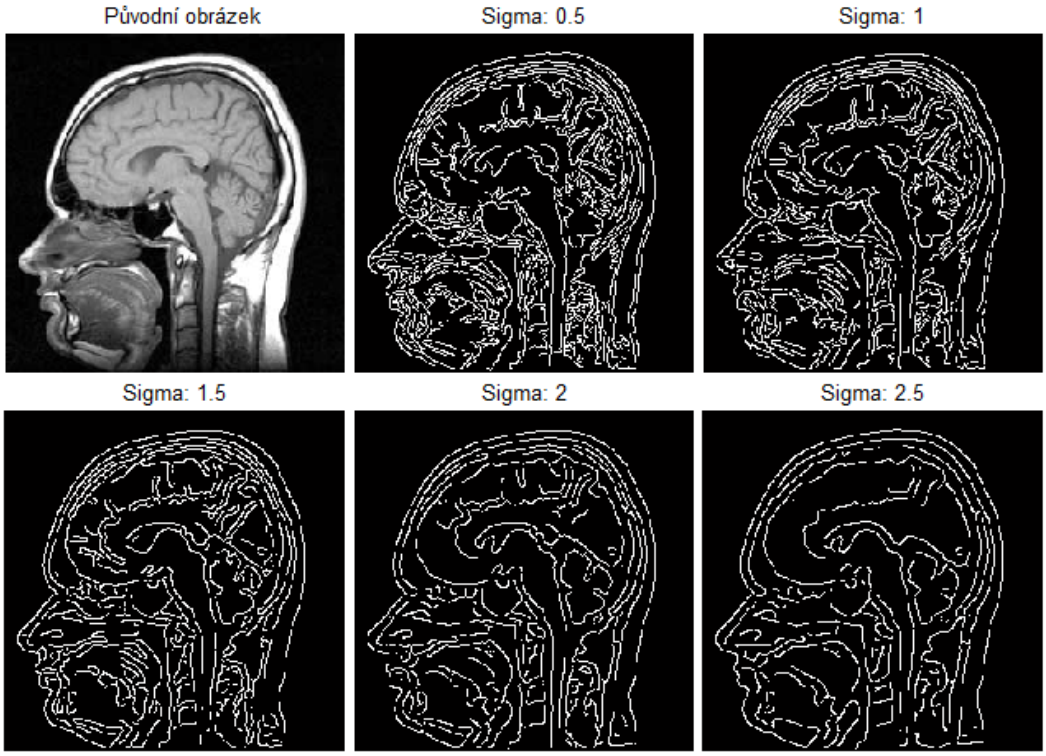


Obr. 48 Aplikace Cannyho detektoru s různým nastavením citlivosti práhu.

Nastavíme-li nulovou prahovou hodnotu, na výsledném snímku se zobrazí nežádoucí šum a neexistující uzavřené obrysy. Po nastavení citlivosti práhu na hodnotu 0,05 se na výsledném snímku šum kolem hlavy na černém pozadí již nevyskytuje, nicméně je velmi obtížné detekovat jakékoliv části snímku. Naopak nastavení citlivosti na hodnotu 0,25 je příliš vysoké, a tím pozvolna ztrácíme informace o jednotlivých hranách a hledaných obrysech, příkladem jsou ztrácející se mozkové závity. Za vhodně nastavenou citlivost práhu považujeme hodnotu 0,15, což je hodnota odpovídající maximální hranici práhu. Minimální hranice odpovídající vhodně nastavené citlivosti je pak hodnota 0,05. Po použití dané citlivosti má možnost uživatel rozpoznat jednotlivé části, jako například míchu. Jelikož není zadána hodnota σ, výsledný snímek ještě není dostatečně zřetelný.

Syntaxe kódu v MATLABu: *BW = edge (I, 'canny', thresh, sigma);*

Uvedená syntaxe specifikuje Cannyho detektor, konkrétněji pak nastavení citlivosti práhu a také hodnotu směrodatné odchylky Gaussova filtru. Výchozím nastavením σ je odmocnina ze dvou, tedy přibližně 1,4. Velikost filtru je zvolena automaticky na základě hodnoty σ.



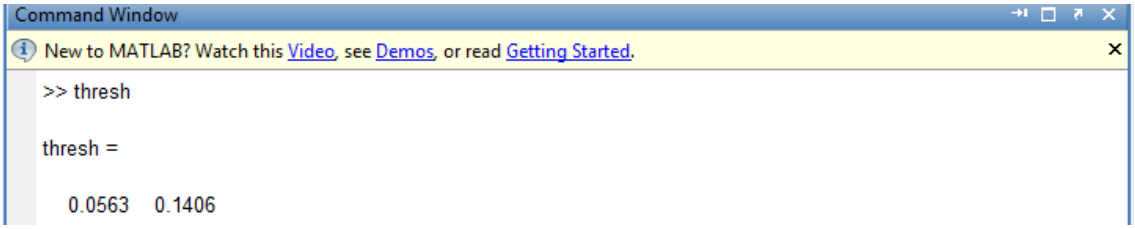
Obr. 49 Aplikace Cannyho detektoru s citlivostí prahu 0,14 a různou hodnotou σ.

Vzorový kód: *cannyho = edge (obr1, 'canny', 0.14, 0.5); cannyho2 =edge (obr1, 'canny', 0.14, 1); cannyho3 = edge (obr1, 'canny', 0.14, 1.5); cannyho4 = edge (obr1, 'canny', 0.14, 2); cannyho5 = edge (obr1, 'canny', 0.14, 2.5);*

Z Obr. 49 můžeme porovnat jednotlivá nastavení hodnoty σ při nastavené citlivosti prahu na 0,14. Daná citlivost je určena ze dvou důvodů. Nejprve je hodnota (0,15) určena za nejvhodněji použitou, viz Obr. 48, a poté je daná hodnota zvolena funkcí *edge*, viz Obr. 50. Nízké nastavení hodnoty σ, mezi které můžeme zařadit hodnoty 0,5 a 1, nejsou účelné, jak je patrné z prvních dvou výsledných snímků. Z těchto snímků lze jen obtížně nalézt potřebné obrysy. Příliš vysoké hodnoty σ, jakými jsou 2,5 a vyšší, jsou taktéž zvoleny nevhodně z důvodu ztrácejících se informací ze snímku. Nejpřijatelnější kombinací nastavení prahu (0,14) a σ považujeme výsledný snímek, který odpovídá nastavení σ mezi 1,5 a 2. Uživatel může rozeznat oddělení míchy od páteře a také přesně ohraničený mozečku při hodnotě σ = 2. Zajímavostí je, že přes všechna jednotlivá nastavení prahu a σ byl u Cannyho detektoru nejpřijatelnějším výstupem první obrázek této kapitoly, viz Obr. 47 , kde byla citlivost prahu a hodnota σ nastavena automaticky.

Syntaxe kódu v MATLABu: *[BW, thresh] = edge (I, 'canny');*

Vzorový kód: *[Cany, thresh] = edge (obr1, 'canny');*



Obr. 50 Příkazové okno s výstupními hodnotami prahování Cannyho detektoru.

Po zadání dané syntaxe získáme z příkazového okna výstup, který odpovídá hranici minimální a maximální hodnoty práhu. To znamená, že ve výsledném snímku se zobrazí veškeré hrany, jejichž hodnota spadá mezi danou hranici. Pro daný snímek jsou zvoleny hodnoty 0,0563 a 0,1406, které jsou považovány za optimální hranici pro detekování hran.

Po rozpracování a následném shrnutí jednotlivých výsledků můžeme říct, že neexistuje optimální nastavení citlivosti práhu ani σ. Jednotlivé snímky se liší stupněm jasu, obrazy jsou více či méně zašedlé, a až podle kvality vstupního snímku můžeme nastavovat dané parametry. Nicméně citlivost práhu se většinou pohybuje v rozmezí hodnot od 0,01do 0,25, s výjimkou metody zero-crossing, kde maximální hodnotou práhu, po které je zobrazen výsledný snímek, je 0,02. Hodnota σ se v této práci pohybuje v rozmezí od 0 po 2,5. Lze použít zdaleka větších hodnot, v našem případě je to zbytečné, protože při nastavení vyšších hodnot než 2,5, se výsledný snímek již nezobrazí.

Výstupem gradientních operátorů je obraz, ve kterém jsou hrany zvýrazněny, nicméně mohou a většinou obsahují další artefakty, se kterými je dále nutné pracovat. Artefakty odpovídají například lokálním nehomogenitám v obraze. Na tyto nepatrné rozdíly je pak možné použít prahování, které je mnohdy postačující. Mezi další artefakty můžeme zařadit přerušované nebo falešné hranice obrazu, které je potřeba dále upravit a zpracovat.

Nelze jednoznačně uvézt, který detektor je pro určitý snímek nejvhodnější, nicméně, jak již víme, nejpropracovanějším detektorem je Cannyho detektor, jemuž také odpovídá výpočetní a tedy i časová náročnost.

5.2 Implementace vlastních funkcí pro detekci hran

Stěžejním úkolem práce je vytvořit algoritmus pro detektory hran, implementovat do prostředí MATLAB a poté srovnat jejich výsledné snímky s přednastavenými detektory. Dalším úkolem je vytvořit grafické závislosti výpočetních časů a opět je porovnat s dostupnými detektory. Implementovány jsou tři detektory, Sobelův, Robertsův a detektor Prewittové. Jsou vytvořeny dva algoritmy pro zmíněné detektory. První typ je vytvořen výpočtem první derivace pomocí diference a druhý pomocí konvoluce. To znamená, že ve výsledku dostaneme šest hranových detektorů – dva pro každý z detektorů. Zmíněné detektory má uživatel možnost porovnat s dostupnými funkcemi v uživatelském rozhraní GUI.

Pro zjednodušení a ukázku, jak fungují implementované detektory pro první typ algoritmu, je vytvořen vývojový diagram. Činnost algoritmu detekce hran pomocí implementovaných detektorů je pro všechny tři detektory stejný, rozdíl je pouze ve výpočetní fázi velikostí gradientu, kde je použit jiný matematický zápis, proto postačí pouze jeden, níže zobrazený diagram.

Postup vytvořeného algoritmu je následující. Po načtení libovolného obrazu, v našem případě medicínského snímku, do příkazového okna MATLABu, je nutné převést snímek na obraz ve stupních šedi, což uděláme pomocí funkce *rgb2gray.*

Obr. 51 Vývojový diagram pro implementované detektory.

Vzorový kód vypadá následovně: *I = rgb2gray (image),*

kde *rgb2gray* je zmíněná funkce, která převádí barevný obraz na stupně šedi a *image* je vstupní proměnná, v našem případě zvolený snímek. Výsledné hodnoty jsou zapsány do proměnné *I*. Abychom mohli s obrazovými hodnotami pracovat, převedeme obraz na datový typ double. Nyní pracujeme s obrazem jako s maticí bodů o stejné velikosti jako je vstupní obraz. Vzorový kód pro převod na double:

I1 = double (I),

kde *I* je již převedený obraz ve stupních šedi, *double* je funkce MATLABu, při které dochází ke konverzi datových typů z uint8 (proměnná *I*) na double, což je nyní proměnná *I1*. Do příkazového okna nyní zapíšeme kód pro implementované detektory, tedy výpočet první derivace pomocí diference, a to jak pro osu x, tak pro osu y. Podívejme se na výpočet pro osu x a y u Sobelova detektoru:

*Gx=((I1(x+2,y)+2*I1(x+2,y+1)+I1(x+2,y+2))-(I1(x,y)+2*I1(x,y+1)+I1(x,y+2)));* *(5.2)*

*Gy=((I1(x,y+2)+2*I1(x+1,y+2)+I1(x+2,y+2))-(I1(x,y)+2*I1(x+1,y)+I1(x+2,y))),* *(5.3)*

kde I1 je vstupní obraz. Po daném výpočtu jsou obrazové body rovny hodnotám Gx (pro osu x) a Gy (pro osu y), které se pohybují v rozmezí od -0,5 po 0,5. Pomocí daných hodnot se vypočítá velikost gradientu, která je dána vztahem:

$$|\nabla f(x,y)| = \sqrt{(Gx^2 + Gy^2)}, \quad (5.4)$$

Pod pojmem porovnávání si představme klasické prahování, které je popsáno v teoretické části, viz vztah (2.2). Výsledná matice má stejné rozměry jako vstupní obraz, ale je tvořena pouze z logických nul a jedniček. Nyní už zbývá jen snímek zobrazit. Nulové hodnoty odpovídají černému pozadí a jedničky pak bílým obrazovým bodům, ze kterých vzniknou jednotlivé detekované hrany a tím i hledané objekty. Vzorový kód pro definování hodnoty prahování je následující:

*prahovani = velikost < thresh*255; velikost (prahovani) = 0;*

Jak vidíme, pokud je po výpočtu hodnota velikosti gradientu nižší než nadefinovaný práh uživatelem, přiřadí se této hodnotě logická nula.

Nyní se podívejme na druhý typ implementovaného algoritmu pro detekování hran pomocí konvoluce. Algoritmus je obdobný algoritmu popsaným výše, liší se jen v drobnostech. Popišme si tedy průběh detekce hran pomocí dané metody podrobněji.

První fází je načtení libovolného snímku uživatelem, který se převede na obraz ve stupních šedi a dále pak na datový typ double, jak je popsáno v předchozím algoritmu. Znovu tedy pracujeme s maticí o určitém rozměru. Do příkazového okna napíšeme masku pro osu x a y jednotlivých operátorů, která je probrána v teoretické části práce. Příkladem můžeme uvést Sobelovu masku, která vypadá následovně:

Sobelova maska pro osu x a y: x = [-1 0 1; -2 0 2; -1 0 1]; y = [-1 -2 -1; 0 0 0; 1 2 1];

Další fází je výpočet hodnot proměnných Gx a Gy pomocí konvoluce. Dané hodnoty potřebujeme pro výpočet velikosti gradientu. Do příkazového okna tedy zapíšeme vztah:

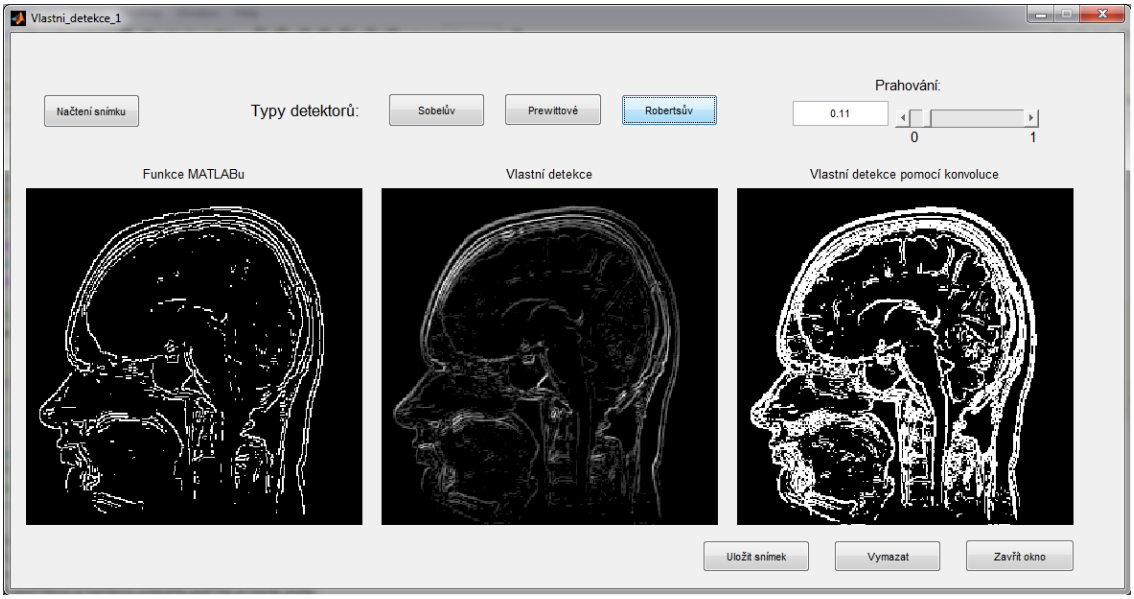
Gx = conv2 (I1, x); Gy = conv2 (I1, y),

kde *conv2* je funkce MATLABu pro výpočet 2D konvoluce, *I1* je matice převedená na datový typ double, *x* a *y* jsou pak zmíněné masky Sobelova detektoru. V této části dochází ke konvoluci původního snímku se Sobelovou maskou zvlášť pro osu x a y. Nyní jsme schopni vypočítat velikost gradientu, která se počítá stejně, jako u předchozího algoritmu, viz vztah (5.4). Poslední částí je prahování a následné zobrazení výsledného snímku.

Obr. 52 Vývojový diagram pro implementované detektory pomocí konvoluce.

Uživatel se dostane do daného prostředí z GUI pro detekci hran. V aplikaci se nachází tři zobrazovací okna, která slouží k zobrazení:

1) přednastavené funkce zvoleného hranového detektoru
2) implementovaného algoritmu pro detekci hran
3) implementovaného algoritmu pro detekci hran pomocí konvoluce.



Obr. 53 Ukázka uživatelského rozhraní pro implementované detektory.

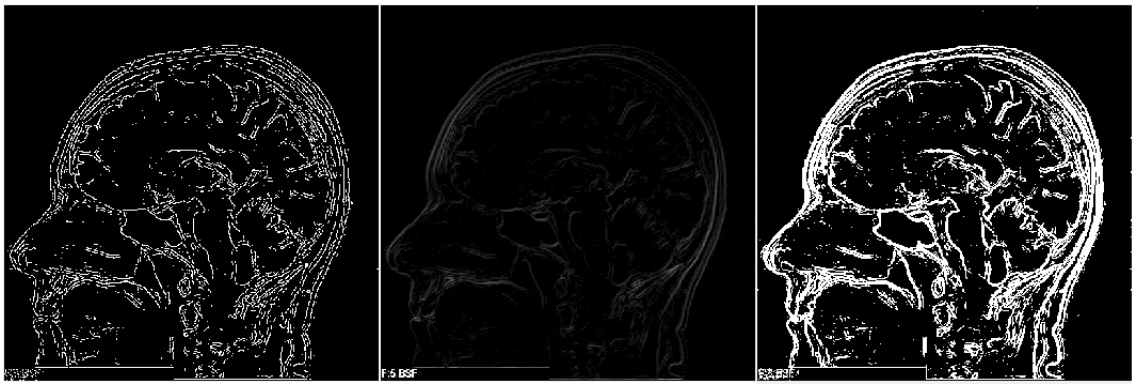
Pro ukázku je načten medicínský snímek mozku, na který je použit Robertsův hranový detektor pro tři různé algoritmy, a to pro funkci *edge* MATLABu, pro implementaci detekce pomocí první derivace a také pro detekci pomocí konvoluce. Jednotlivé výstupní snímky se výrazně liší. Pro začátek je nutné říct, že nastavená prahová hodnota není optimální pro všechny výsledné snímky. Prostředí GUI je vytvořeno pro porovnání výstupních snímků jednotlivých algoritmů. Pro každý dílčí algoritmus je optimální jiná hodnota citlivosti práhu na zvolený snímek. V případě, že uživatel najede myší na výstupní obraz, je možné si jej pomocí lupy přiblížit.

Obr. 54 Sobelův detektor. Zleva: funkce MATLABu, implementovaná funkce, implementace pomocí konvoluce.

Na obrázku jsou zobrazeny tři výsledné snímky pro různé algoritmy použití Sobelova detektoru a odlišných hodnot práhu. Citlivosti práhu jednotlivých výstupních snímků jsou nastaveny pro uživatele co nejvhodněji. Pro funkci MATLABu je nastavena hodnota práhu 0,049, pro implementovanou funkci 0,43, a pro implementovanou funkci pomocí konvoluce 0,0455. Z výsledného snímku po aplikování funkce MATLABu je patrné, že daná funkce disponuje s dalšími funkcemi pro vylepšení, mezi které patří funkce *thinning*, která je vysvětlena v předchozích kapitolách, viz strana 32. Detekované hrany jsou podstatně užší v porovnání s implementovanými detektory. Na nich mohou vznikat nepřesně detekované obrazové body, které neodpovídají vstupnímu snímku.

Obr. 55 Detektor Prewittové. Zleva: funkce MATLABu, implementovaná funkce, implementace pomocí konvoluce.

Na dalším snímku jsou použity algoritmy pro detekci pomocí operátoru Prewittové. Hodnoty citlivosti práhu pro optimální nastavení výsledných snímků se i v daném případě liší. Pro funkci MATLABu je nastaven práh na hodnotu 0,08, pro implementovanou funkci na hodnotu 0,43, a pro implementovanou funkci s použitím konvoluce je nastavena citlivost práhu na hodnotu 0,52. Přesto, že nastavení práhu je co nejvhodnější, na snímcích se vyskytuje šum kolem detekovaných hran. Výhodou implementovaných algoritmů pro detektor Prewittové je pro zvolený medicínský snímek nalezený výrazný objekt ve středu krční páteře. Můžeme říct, že výsledný snímek odpovídající použití algoritmu pomocí konvoluce je optimální, nežádoucí šum se vyskytuje pouze v malém množství a hrany jsou téměř nepřerušované. Jedinou nevýhodou je zmíněná nepřesnost lokalizace hranových bodů způsobená širokými hranami.



Obr. 56 Robertsův detektor. Zleva: funkce MATLABu, implementovaná funkce, implementace pomocí konvoluce.

Posledním z implementovaných detektorů je Robertsův detektor, pomocí kterého jsou vytvořeny algoritmy k detekci hran. Citlivost práhu je i v tomto případě nastavena co nejvhodněji. Pro ukázku výsledků je opět použit jiný medicínský snímek, na kterém poukážeme rozdíl při nastavení prahové hodnoty. Pro funkci MATLABu je nastavena hodnota práhu 0,035, pro implementovaný detektor 0,045 a pro implementovaný detektor pomocí konvoluce 0,053. Můžeme usoudit, že vstupní snímek je velmi tmavý, a proto je nutné nastavit citlivost práhu na velmi nízké hodnoty. Přesto je velice těžké detekovat na daném snímku jednotlivé hrany. Na prvním snímku jsou výsledné hrany nejasné a přerušované. Druhý snímek je velmi tmavý a detekované hrany nelze téměř vidět. Jedinou výhodou dané detekce je přesné detekování tvaru lebky spojenými hranami. Nalezené hrany na posledním jsou nejvýraznější s občasným výskytem nežádoucího šumu kolem hran.

Z výsledných snímků usuzujeme, že nastavená citlivost práhu se pro optimální výsledek značně liší. Jedním z důvodů je algoritmus, který je pro přednastavenou funkci propracovanější. Výpočetní čas algoritmu je rychlejší, a navíc jsou v dané funkci zakomponovány vnitřní funkce, které mají za úkol například zúžit detekované hrany. Tyto funkce v implementovaných algoritmech zakomponovány nejsou. Dalším důvodem je, že jednotlivé vstupní snímky jsou odlišné, jsou více či méně tmavé, což znamená, že s každým vstupním snímkem se citlivost práhu liší. Proto je nutné, aby uživatel reagoval na jednotlivý vstupní snímek zvlášť, a na jeho základě teprve nastavil optimální hodnotu práhu. Jasným příkladem jsou použité hodnoty práhu pro implementované algoritmy, viz Obr. 55 , které jsou 0,43 a 0,52 v porovnání s Obr. 56 , kde jsou pro implementované algoritmy použity hodnoty 0,045 a 0,053.

5.2.1 Srovnání výpočetních časů

Pro porovnání výpočetních časů přednastavených detektorů s implementovanými je po naměření jednotlivých hodnot vytvořena grafická závislost výpočetních časů na různé velikosti medicínského snímku. Jelikož se pohybujeme v jednotkách milisekund a každý počítač má jiné technické dispozice, výpočetní časy se budou lišit.

Výsledné hodnoty nejsou absolutní, časy můžou být rychlejší nebo pomalejší. Cílem není poukázat na rychlost výpočtu, ale na časové porovnání výpočtu dostupných detektorů s implementovanými, pro různé rozměry snímku. Pro názornou ukázku a porovnání jsou výsledky dostačující.

Tab. 1 Porovnání výpočetních časů pro Sobelův detektor.

Počet pixelů (N)	Funkce MATLABu – t1(ms)	Implementovaná funkce – t2(ms)
256x256	230,158	486,147
231x231	175,814	401,824
205x205	101,28	319,104
180x180	85,342	245,669
154x154	61,862	179,463
128x128	44,054	126,087
103x103	26,586	80,241
77x77	16,42	46,713
52x52	6,237	20,311
26x26	2,234	4,359

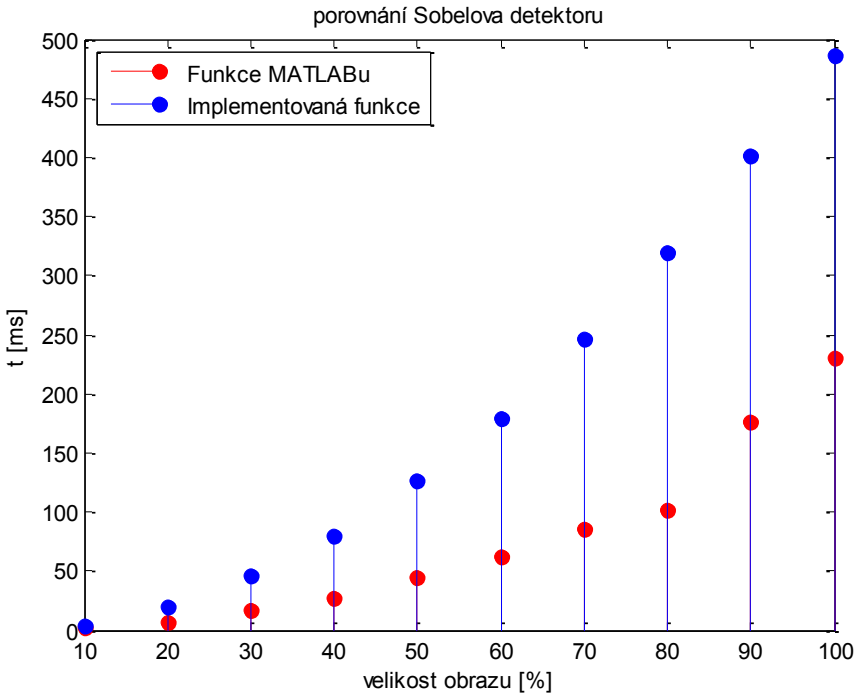
V tabulce jsou zobrazeny jednotlivé výpočetní časy dostupného a implementovaného detektoru, a k nim odpovídající rozměry snímků. Pro zvolený snímek je pomocí funkce *imresize* vytvořeno deset rozměrů. Největší snímek odpovídá zvolenému, tedy rozměru 256x256 pixelů, a další snímky jsou vytvořeny s postupným 10% zmenšením. Pro porovnání získáme 10 snímků různé velikosti. Příklad funkce *imresize* vypadá následovně:

A = imresize (I1, 0.90),

kde *imresize* je funkce MATLABu pro změnu velikosti obrazu, *I1* je zvolený vstupní obraz, hodnota 0,90 odpovídá procentuálnímu zmenšení výstupního obrazu *A*. Výsledkem je tedy zmenšený obraz o 10 %. Výpočetní časy jsou změřené pomocí funkce časovače *tic* a *toc*.

Vzorový příklad: *tic sobel_90 = edge (A, 'Sobel', 0.10); toc,*

kde *tic* je počáteční stav měření výpočtu, *toc* je ukončení tohoto měření, *sobel_90* má rozměr 90 % původního obrazu a je to proměnná, do které se uloží detekce hran pomocí funkce *edge* Sobelovým detektorem, s nastavenou citlivostí práhu na hodnotu 0,1.



Graf č. 1 Porovnání výpočetních časů pro Sobelův detektor.

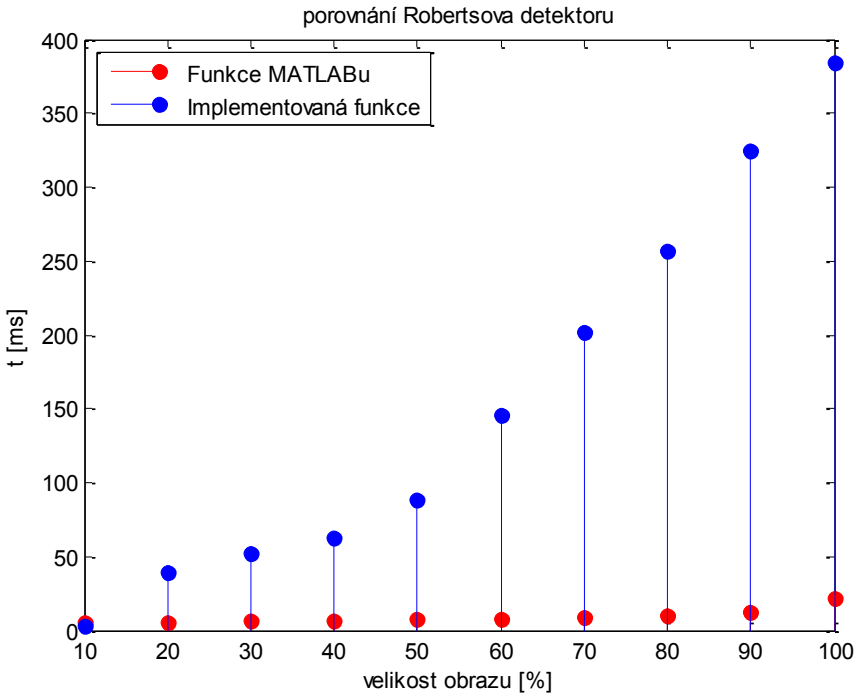
Na základě naměřených hodnot z Tab. 1 je vynesen diskrétní graf závislosti času t (ms) na velikosti obrazu vyjádřené v procentech. Na vertikální ose je znázorněno procentuální zmenšení daného obrazu. Na horizontální ose je znázorněn čas, jehož maximální hodnota je 500 ms. Modře vyznačené body odpovídají implementovanému algoritmu a červené body dostupné funkci pro Sobelův detektor.

Z grafu vyplývá, že výpočetní časy pro přednastavený detektor jsou více než dvakrát, některé i třikrát rychlejší ve srovnání s implementovaným algoritmem. Výpočetní časy implementované funkce jsou úměrné rozměru obrazu, naopak u funkce MATLABu se vyskytují lehké výchylky, které ovšem nemají žádný velký význam. Výpočetní čas daného implementovaného algoritmu pro originální obraz je téměř půl sekundy, na rozdíl od dostupné funkce, která trvá pouze 230 ms.

Tab. 2 Porovnání výpočetních časů pro Robertsův detektor.

Počet pixelů (N)	Funkce MATLABu – t1(ms)	Implementovaná funkce – t2(ms)
256x256	22,096	384,77
231x231	12,203	324,544
205x205	9,726	256,945
180x180	8,189	202,103
154x154	7,397	145,822
128x128	7,319	88,612
103x103	6,376	62,279
77x77	6,11	52,225
52x52	5,197	39,056
26x26	4,932	3,124

Z naměřených výpočetních časů pro Robertsův detektor pozorujeme, že se výsledné hodnoty jednotlivých algoritmů výrazně liší. Výpočetní časy funkce MATLABu se pohybují v rozmezí od 4 do 22 ms, zatímco implementovaná funkce má větší časový rozptyl, od 3 do 384 ms. Pomocí funkce MATLABu je výpočet algoritmu originálního snímku sedmnáctinásobně rychlejší.



Graf č. 2 Porovnání výpočetních časů pro Robertsův detektor.

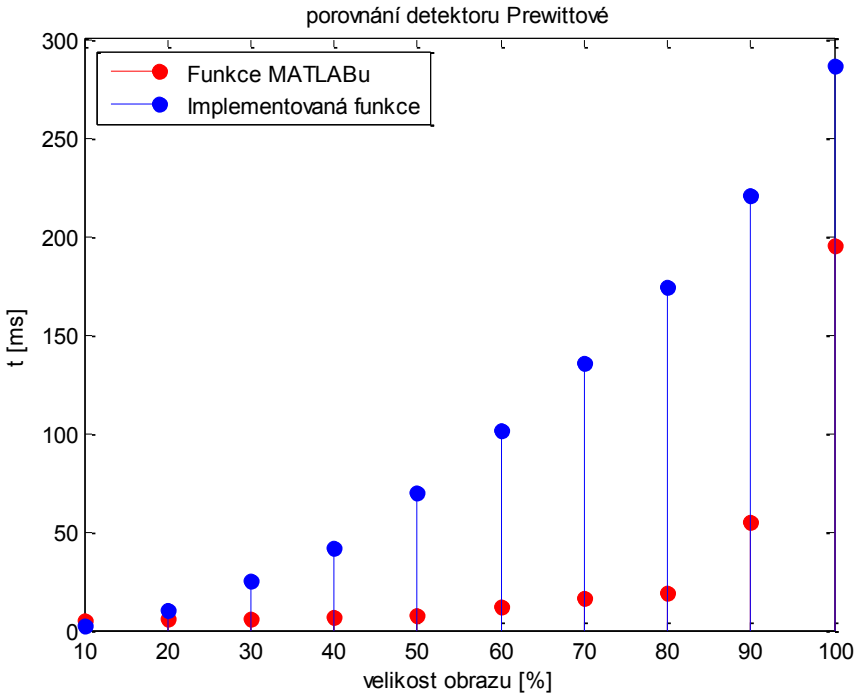
Výpočetní časy implementovaných algoritmů pro Robertsův a Sobelův detektor jsou téměř shodné. Robertsův detektor má však ve většině případů pro odlišné rozměry obrazů rychlejší výpočetní časy. Například výpočet originálního snímku pro daný detektor je o 100 ms rychlejší než pro Sobelův detektor, viz Tab. 1.

Zaměřme se na výpočetní časy funkce MATLABu pro Robertsův detektor. Z Grafu č. 2 je evidentní, že rozměr obrazu nemá žádný vliv na výpočetní časy Robertsova detektoru. Hodnoty se pohybují maximálně kolem 20 ms, což je v některých případech výpočetní čas 20% velikosti obrazu.

Tab. 3 Porovnání výpočetních časů pro detektor Prewittové.

Počet pixelů (N)	Funkce MATLABu – t1(ms)	Implementovaná funkce – t2(ms)
256x256	194,988	286,347
231x231	54,941	220,847
205x205	18,931	174,348
180x180	15,931	135,511
154x154	11,958	101,414
128x128	7,062	69,595
103x103	6,515	41,865
77x77	6,03	24,798
52x52	5,662	10,43
26x26	5,22	2,407

Jednotlivé časy nejsou tak úměrné, jako v předchozích tabulkách. Dokonce se zde nachází jediná hodnota výsledného času, která je pro implementovaný algoritmus kratší než pro dostupnou funkci v MATLABu. Co se týče výpočetní rychlosti implementovaných algoritmů jednotlivých detektorů, je detektor Prewittové ze všech nejrychlejší. Nejkratší čas je nejen pro nejmenší velikost obrazu, což je 2,4 ms, ale také pro originální obraz, jemuž odpovídá výpočetní čas 286 ms. Výpočetní časy funkce MATLABu jsou velmi nevyvážené. Pro šest velikosti obrazů se časy pohybují v rozmezí od 5 do 15 ms, ostatním odpovídá předpoklad, že s rostoucí velikostí obrazu roste i výpočetní čas. U veškerých velikostí obrazu pro dostupnou funkci je výpočetní čas do 55 ms, pouze originální obraz je vypočten s téměř čtyřnásobným zpožděním.

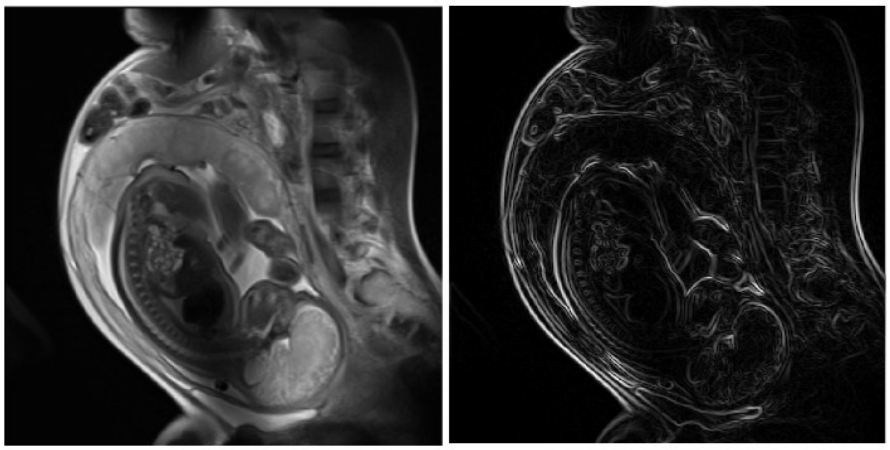


Graf č. 3 Porovnání výpočetních časů pro detektor Prewittové.

Z daného grafu vyplývá, že výpočetní časy implementovaného algoritmu pro detektor Prewittové je opět úměrný. Na horizontální ose postačí maximální hodnota 300 ms, výpočetní čas originálního obrazu je 286 ms.

Rychlost výpočtu přednastavených funkcí pro detekci hran je jednoznačně rychlejší jak při různých velikostech, tak u jednotlivých hranových detektorů. Je to způsobeno tím, že funkce MATLABu disponují s dalšími vnitřními funkcemi, které výpočet algoritmu zrychlují.

Závěrem můžeme říct, že u detektorů Sobelova a Prewittové, ať dostupných, nebo implementovaných, se zvyšujícím se rozměrem obrazu roste i jeho výpočetní časová náročnost, viz Graf č. 1 a Graf č. 3. Tento jev jsme mohli předpokládat, nicméně značný rozdíl pozorujeme u Robertsova detektoru, konkrétně u přednastavené funkce, kde je výpočetní čas při jakémkoliv rozměru obrazu téměř konstantní. Důvodem je rozměr masky Robertsova operátoru (2x2 pixelů), která je nejjednodušší a nejrychlejší na výpočet. Jak vidíme, velikost obrazu nemá žádný vliv na výpočetní čas pro daný hranový detektor, viz Graf č.2.



Obr. 57 Zleva: Původní snímek, implementovaný algoritmus.

Světlejší snímky se lépe detekují pomocí přednastavených funkcí, nicméně velmi tmavé snímky, viz Obr. 56 jsou téměř nepoužitelné. Výsledný snímek je zrnitý s nadměrným obsahem nežádoucího šumu. Pro příliš tmavé snímky, jako zmíněný z Obr. 56 a nyní daný vstupní snímek plodu v pozdní fázi těhotenství, je vhodným právě implementovaný algoritmus pro detekci hran. Hrany nejsou tak výrazné, nicméně jsou velmi přesné, spojité a bez vyskytujícího se šumu. Na výstupním snímku můžeme spolehlivě a přesně detekovat tvar plodu a další zájmové objekty. Na daný snímek je aplikován algoritmus pro detekci hran pomocí detektoru Prewittové s nastavenou citlivostí práhu na hodnotu 0,009. Dalším příkladem může být snímek CT mozku, jehož výstup je uveden v příloze, viz Obr. 69 .

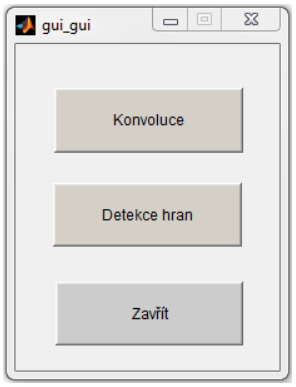
5.3 Grafické uživatelské rozhraní

V prostředí MATLAB je vytvořeno grafické uživatelské rozhraní (využijeme zkratku GUI) pro jednotlivé funkce. Aplikaci lze ovládat pomocí jednotlivých interaktivních prvků.

Nyní si představme vývojový diagram celkového grafického uživatelského rozhraní. Základem je vstupní GUI, pomocí kterého má uživatel možnost výběru k přístupu do dalšího uživatelského rozhraní pro konvoluci nebo detekci hran. V prostředí pro detekci hran je možnost volby použití jednotlivých detektorů a také přístup do dalších dvou rozhraní. Veškeré výstupy medicínských snímků po aplikování jednotlivých funkcí, ať už dostupných nebo implementovaných, jsou zobrazeny a porovnány se vstupním snímkem, tomu odpovídají zelená zobrazovací okna na konci každého z výstupů, viz Obr. 68 .

5.3.1 Uživatelské rozhraní pro konvoluci

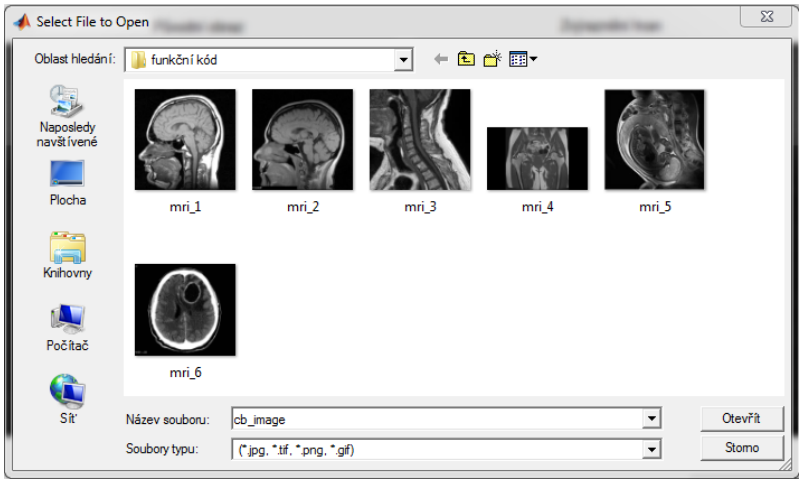
Kapitola se věnuje implementaci filtrů na základě konvoluce v prostředí MATLABu. Nyní si představme vstupní uživatelské rozhraní.



Obr. 58 Vstupní uživatelské rozhraní.

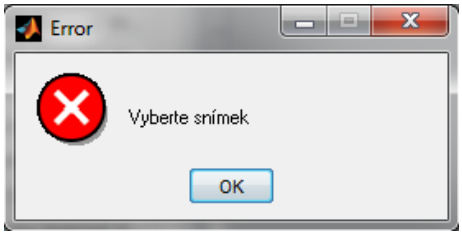
Jestliže uživatel zvolí možnost konvoluce, otevře se nové uživatelské rozhraní, viz Obr. 61. Prostředí obsahuje jednotlivé typy filtrů, které se aplikují na zvolený medicínský snímek. Existuje spousta filtrů využívané ke konvoluci, pro ukázku jsou zvoleny pouze čtyři nejznámější. Původní a výstupní snímek se pro porovnání vyskytují v jednotlivých zobrazovacích oknech. Horní lišta rozhraní obsahuje tlačítko help, ve kterém jsou rozebrány jednotlivé filtry a jejich vzorové kódy. Jeho úkolem je pomoci uživateli zorientovat se mezi jednotlivými filtry, slouží tedy jako jakási stručná uživatelská příručka.

Prvním krokem uživatele je načíst snímek, po využití tlačítka pro načtení se zobrazí dialogové okno, ze kterého si může uživatel vybrat z jednotlivých medicínských snímků.



Obr. 59 Ukázka okna pro výběr medicínského snímku.

V případě, že uživatel snímek nevybere, je aplikace ošetřena různými podmínkami, jako například dané chybové hlášení, které napíše, že je nutné si z nabízejících snímků zvolit.



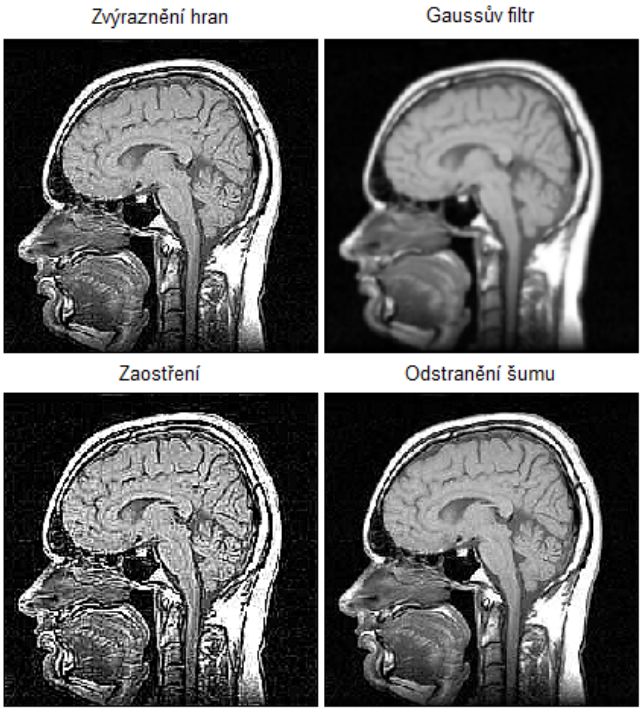
Obr. 60 Ukázka výstražného okna.

Celková aplikace je ošetřena podmínkami tohoto typu. Například u detekce hran jsou ošetřeny podmínky pro nutnost nastavení citlivosti prahu nebo jeho maximální rozsah, nebo také nastavení hodnoty σ (pro určité detektory).

Vraťme se k vytvořené aplikaci pro konvoluci a k možnostem použití jednotlivých filtrů. Je použito čtyř různých filtrů pro představu uživatele, jaký vliv mají filtry na zvolený snímek. Po načtení snímku se zvolí určitý filtr a provede se výpočet konvoluce, který je realizován uvnitř zvolené funkce. Není to nic jiného, než aplikování jedné z konvolučních masek postupně skrz celý obraz, takzvané plovoucí okno. Zmíněný proces je podrobně vysvětlen v teoretické části, viz Obr. 8. Po aplikování dané masky na původní snímek se změní hodnoty jeho obrazových bodů, a tímto se také změní výstupní snímek.

Obr. 61 Uživatelské rozhraní pro konvoluci s použitím zvýraznění hran.

Pro ukázku je načten snímek a použita konvoluční maska, která má za úkol zvýraznit hrany vstupního snímku, v našem případě na snímku z magnetické rezonance. Povšimněme si, že hrany jsou zřetelnější a celkový výstupní snímek je výraznější než původní.



Obr. 62 Souhrn filtrů použitých ke konvoluci.

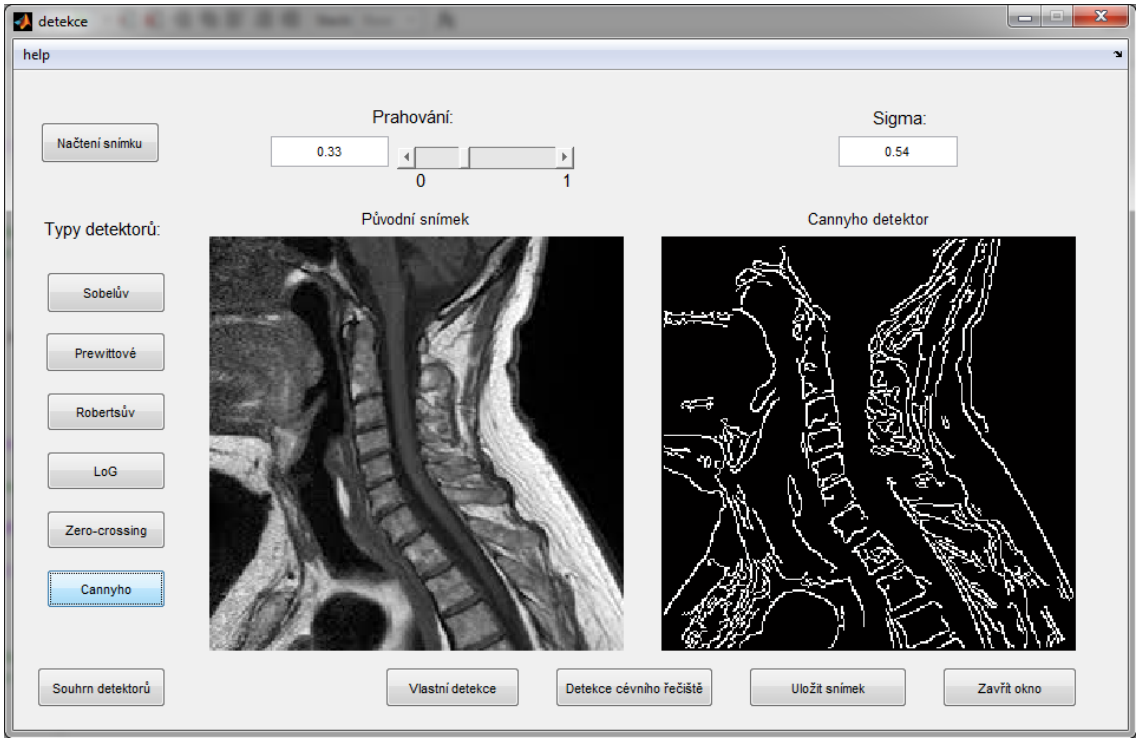
Na obrázku je zobrazen souhrn použitých filtrů z GUI konvoluce. Filtr pro zvýraznění hran je okomentován výše, viz Obr. 61 . Přejděme tedy ke Gaussovu filtru, který se využívá především k vyhlazení a k odstranění šumu v obraze. Po použití konvoluce a Gaussova filtru se šum ze snímku vytratí, nicméně dochází k poměrně značnému rozmazání, což je nežádoucí jev vedoucí k dalším problémům při segmentaci, např. u detekce hran. Po aplikaci konvoluční masky pro zaostření si můžeme povšimnout zřetelného zhoršení výstupního snímku. Daný snímek se může jevit jako jasnější a výraznější, avšak jeho kvalita výrazně klesá, dochází zde k degradaci obrazové kvality a také k výskytu nežádoucího šumu. Důvodem je fakt, že vyšší frekvence nesou informace o hranách spolu se šumem. Proto je nutné správně rozhodnout, jak snímek upravit, aby došlo ke korektnímu zvýraznění vlastností daného snímku. Příkladem zvýrazněných hran můžeme uvézt zářezy mozkové kůry, které jsou na výstupním snímku v porovnání s původním nápadně tmavší, stejně jako ohraničení mozkomíšního moku. Poslední snímek odpovídající odstranění šumu považujeme za nejlepší výstup z použitých filtrů. Snímek je optimálně vyvážen, není příliš přeostřen ani naopak rozmazán a filtrem je odstraněn šum, který se již vyskytuje v minimálním množství.

5.3.2 Uživatelské rozhraní pro detekci hran

Prostředí obsahuje dostupné hranové detektory funkce *edge* MATLABu. Při konvoluci se aplikuje konvoluční maska na určitý snímek, ale při detekci hran je nutné udělat výpočet pomocí první nebo druhé derivace, což má za výsledek odpovídající časovou náročnost.

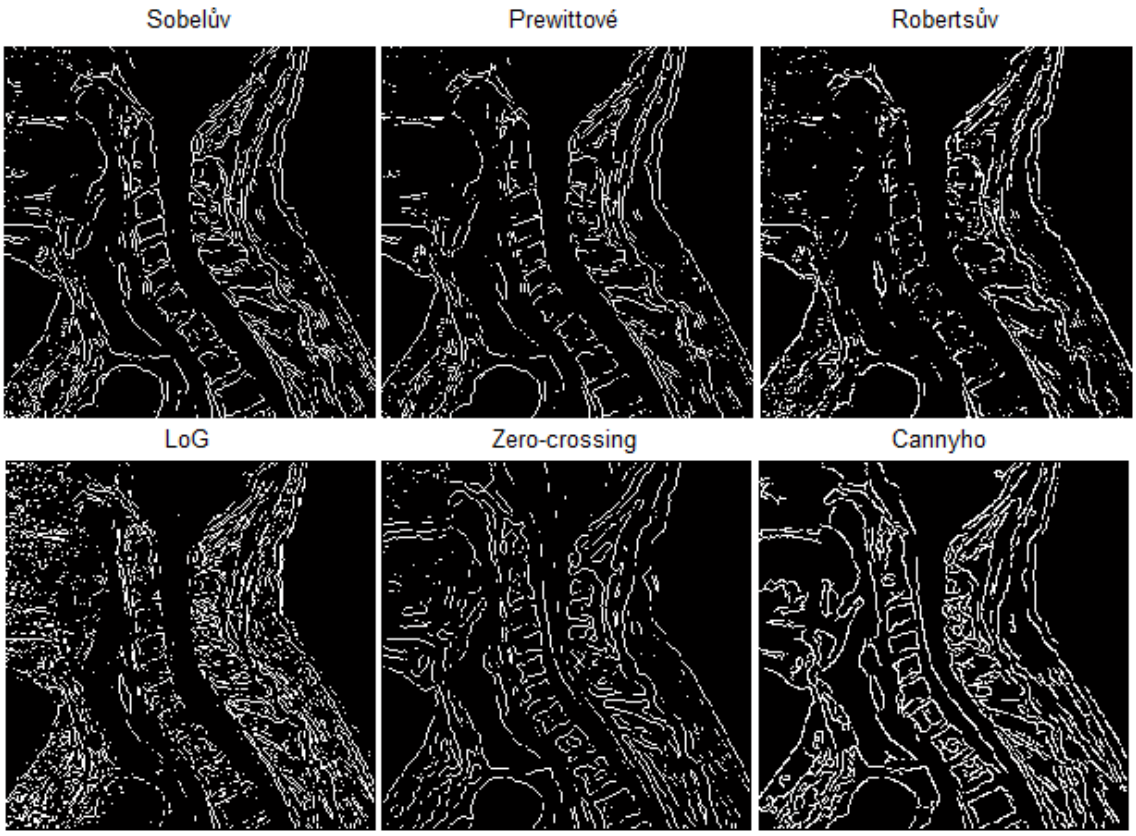
V případě, že detekujeme hrany, můžeme nastavit hodnotu citlivosti práhu (0–1) a také σ, používané hodnoty jsou v rozmezí od 0,5 do 3. Jestliže si uživatel nezvolí snímek, nenastaví hodnotu práhu nebo σ, zobrazí se výstražné okno stejně jako u konvoluce, aby tak učinil, viz Obr. 60 . Uživatelské rozhraní obsahuje zobrazovací okna pro porovnání původního snímku s výsledným. Výsledný snímek je možno přiblížit pokud uživatel najede myší na daný snímek.

Veškeré dostupné hranové detektory funkce edge a jejich možnosti nastavení parametrů jsou rozebrány v kapitole 5.1 na straně 29, a proto je nyní pouze představeno uživatelské rozhraní dané problematiky. V aplikaci se rovněž vyskytuje tlačítko help sloužící uživateli jako pomoc pro vysvětlení a zorientování se mezi jednotlivými detektory.



Obr. 63 Uživatelské prostředí pro detekci hran.

Na obrázku se nabízí všechny přednastavené detektory funkce *edge*. Pro příklad je zvolen medicínský snímek mozku v sagitální rovině z vyšetření magnetickou rezonancí a je použit Cannyho hranový operátor, který je ze všech detektorů nejpropracovanější. Na výsledném snímku je nastavena citlivost práhu na hodnotu 0,33 a σ odpovídá hodnotě 0,54.



Obr. 64 Souhrn použitých detektorů na snímek krční páteře.

Nyní máme možnost porovnat výstupní snímky všech šesti přednastavených detektorů. Jednotlivé parametry (práhu a σ) jsou u všech detektorů nastaveny tak, aby byl výstupní snímek pro uživatele co nejužitečnější. Nastavená citlivost práhu se pohybuje v rozmezí od 0,02 po 0,3 a hodnota σ pak od 0,5 až 2. Jak vidíme, nejlepšími výslednými snímky jsou Cannyho detektor a Prewittové, které obsahují velmi malé množství nežádoucího šumu. Naopak značné množství šumu je obsaženo po použití detektoru LoG. Mezi nevýhody výsledného snímku po aplikaci detektoru LoG a Zero-crossing patří vyskytující se přerušované hrany.

Jestliže chceme porovnat detailní změny na jednotlivých snímcích, můžeme si je přiblížit pomocí lupy. V každém případě je nutné pro absolutní detekci následně použit dalších filtrů pro upravení daného snímku. Pro další ukázku porovnání je použit snímek mozku z magnetické rezonance, na kterém jsou opět použity všechny hranové detektory s co nejvhodnějším nastavením parametrů. Daný výstup nalezneme v příloze, viz Obr. 71 .

5.4 Detekce cévního řečiště oka

Poslední kapitola je vytvořena pro praktickou ukázku využití detektorů hran k segmentaci snímků, konkrétně pak v očním lékařství. Cílem je detekovat cévní řečiště a optický nerv pomocí vhodného hranového detektoru. Segmentace je provedena pomocí Kirschova operátoru, který je založen na výpočtu gradientu pomocí aproximace první derivace. Jeho vývojový diagram vypadá následovně:

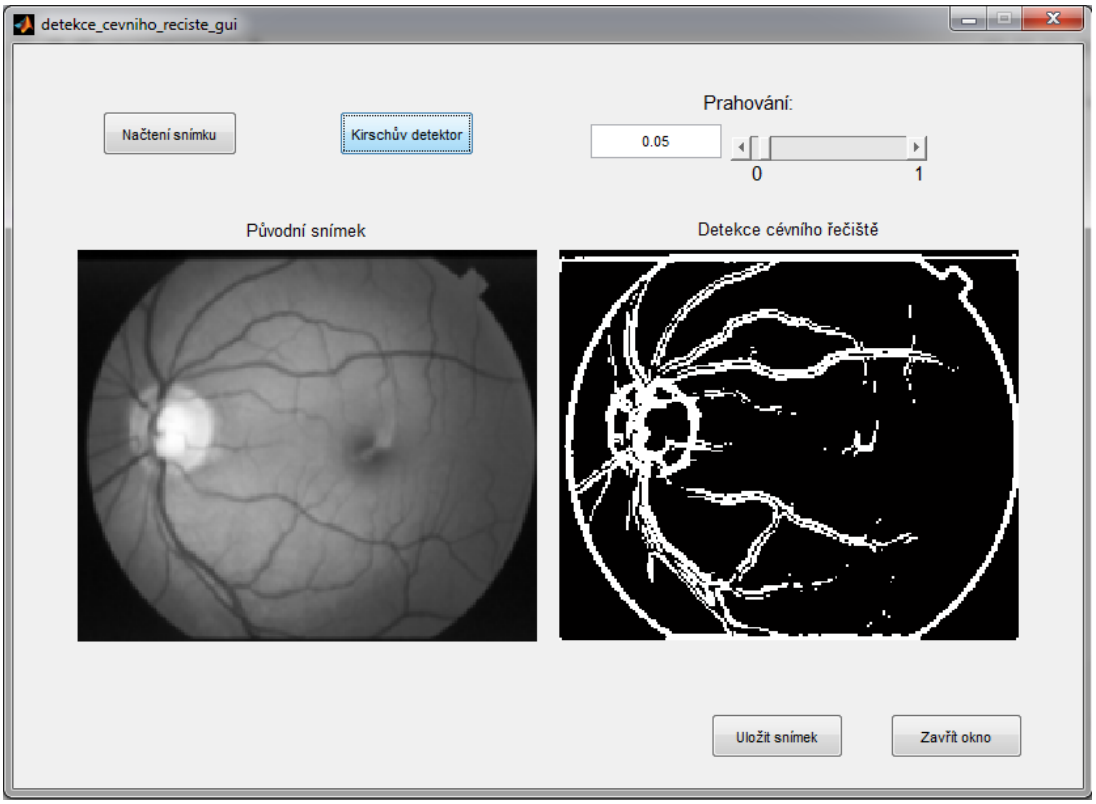
Obr. 65 Vývojový diagram pro detekci cévního řečiště oka Kirschovým detektorem.

Zvolený snímek uživatelem je vstupní proměnnou, což jsou obrazová data převedená na obraz ve stupních šedi. V této fázi se obraz proloží jednotlivými maskami Kirschového operátoru, viz (4.28) z teoretické části. Masky se aplikují postupně pro všech osm směrů. Nyní se provede filtrace pro jednotlivé masky se vstupním obrazem. K realizaci filtrace je využita funkce *filter2*, která má za úkol filtrovat data použitím 2D korelace.

Vzorový kód: *t1= filter2 (h1, inImg),*

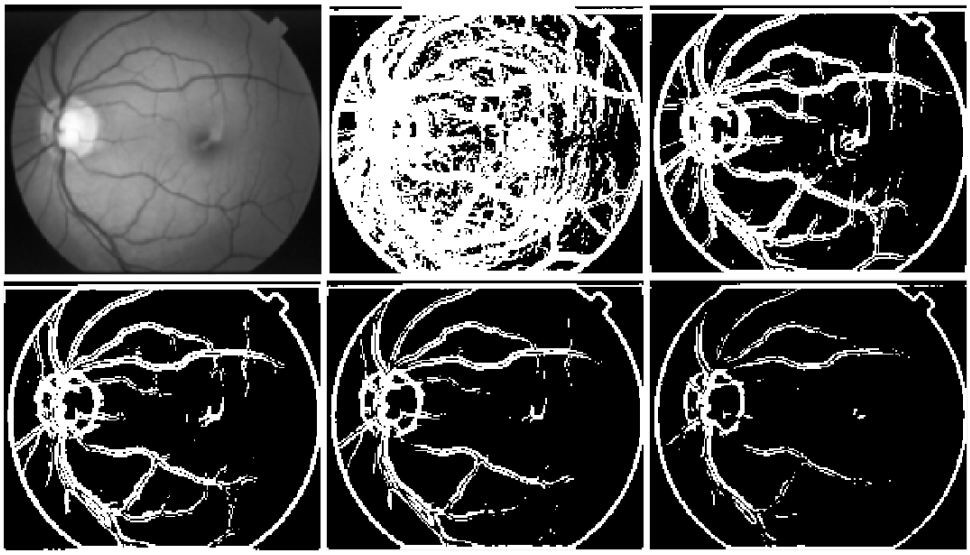
kde *t1* je výstupní segmentovaný obraz, *filter2* je dostupná funkce MATLABu, *h1* je jedna z osmi možných použitých směrů použitá a *inImg* je vstupní proměnná. Výsledkem je matice obrazových bodů o stejném rozměru jako vstupní proměnná. Nyní získáme osm nových proměnných, ve kterých se hledá největší velikosti gradientu. Na odpovídající matici (největších hodnot gradientu) je proložena vhodná citlivost práhu nastavená uživatelem. Výsledkem je binární obraz tvořen z logických jedniček a nul.

Po otevření okna pro detekci cévního řečiště (z GUI pro detekci hran) se zobrazí uživatelské rozhraní, které nám poskytuje funkci načtení snímku a dvě zobrazovací okna. Po nastavení hodnoty práhu se zobrazí výstupní snímek, ve kterém se použitím Kirschova operátoru vysegmentuje oční pozadí a vyextrahuje tvar a průběh cév. Dalším cílem je detekovat zrakový nerv. Snímek je možný přiblížit pomocí lupy a případně jej uložit pro následné využití.



Obr. 66 Ukázka uživatelského prostředí pro detekci cévního řečiště.

Na obrázku níže je zobrazeno pět různých citlivostí práhu použitých snímek oka v pořadí od nejnižší po nejvyšší hodnotu. Použité hodnoty práhu jsou velmi nízké a při změně citlivosti pouze o jednu setinu jsou velmi výrazné změny ve výstupních snímcích.



Obr. 67 Vliv citlivosti práhu na výsledný snímek. Zleva: Původní snímek, práh: 0,01; 0,03; 0,04; 0,05; 0,07.

Pro ukázku nesprávného nastavení práhu je použita hodnota 0,01. Nežádoucí šum, který je propuštěn a zobrazen z důvodu příliš nízké nastavené hodnotě. Citlivost práhu 0,03 je výrazně lepší, výhodou je, že při velmi nízké hodnotě práhu lze detekovat i drobnější cévky. Nevýhodou je pak vyskytující se množství šumu. Poslední snímek odpovídající práhu 0,07 také není zcela optimální. Povšimněme si bílého pásu vyskytujícího-se v horní části výsledného snímku. Pás je nežádoucí šum, který se zvyšující se hodnotou práhu postupně zužuje a ztrácí. Po použití citlivosti práhu 0,07 na zvolený snímek již není pás téměř vidět, nicméně se ze snímku ztrácí i žádoucí hrany, v našem případě cévy. Ideálním nastavením práhu pro zvolený snímek sítnice jsou hodnoty 0,04 nebo 0,05, při kterém je nejlépe znázorněn optický nerv a průběh jednotlivých cév.

Nevýhodou použití Kirschova detektoru jsou vyskytující-se artefakty odpovídající zdvojeným hranám na výsledném snímku a také obsažený šum kolem detekovaných hran. Uvedené nevýhody se bohužel vyskytují u všech detektorů, tyto problémy se řeší filtrací a následnými úpravami. Pro zajímavost je na snímek oka použit implementovaný Sobelův detektor uvedený v příloze, viz Obr. 70 .

Cílem optimální detekce je pár základních bodů, které si shrneme. Jedním z nejdůležitějších je správná detekce. Pod tímto pojmem si představme detektor, který má za úkol minimalizovat pravděpodobnost falešných hran způsobených nežádoucím šumem stejně jako minimalizovat ztrátu opravdových hran. Dalším bodem je správná lokalizace, což znamená, že detekované hrany se vyskytují ve výsledném snímku tak, aby co nejpřesněji odpovídaly hranám vyskytujícím se v původním snímku. Poslední bod zahrnuje pouze jedinou odpověď na reálné hranové body, to znamená minimalizovat počet lokálních maxim kolem reálné hrany.

Závěr

Cílem této bakalářské práce bylo zaměřit se na problematiku segmentace obrazu, konkrétně pak medicínských snímků. Segmentaci jsem realizovala na základě dostupných hranových detektorů funkce edge v MATLABu, a také pomocí vlastních implementovaných algoritmů. V práci jsem podrobně rozebrala přednastavené funkce MATLABu s jejich možnostmi nastavení parametrů a filtry, které mohou být použity ke konvoluci.

Implementovala jsem tři detektory hran na základě výpočtu první derivace pomocí diference, a tři detektory pomocí konvoluce. K implementaci detektorů jsem použila masky pro Robertsův, Prewittové a Sobelův detektor. Algoritmy jsou vysvětleny pomocí jednotlivých vývojových diagramů.

Obsahem práce jsou výpočetní časy dostupných a implementovaných detektorů, které jsou pro porovnání uvedeny v tabulkách a vykresleny v grafech. Z výpočetních časů je zřejmé, že algoritmy přednastavených funkcí jsou jednoznačně rychlejší.

Výstupem je grafické uživatelské rozhraní, které slouží jako simulační úloha pro pochopení principu detekce a vlivu jednotlivých detektorů na daný snímek. Výstupní snímky jsou vykresleny v zobrazovacích oknech GUI v jednotlivých kapitolách, nebo v příloze. Detekci cévního řečiště a zrakového nervu jsem uskutečnila pro zajímavost a praktické využití v očním lékařství.

Práce slouží především ke studijním účelům a pro zorientování se mezi jednotlivými detektory hran. Může být rozvinuta o další algoritmy pro detekci hran, popřípadě zaměřena na filtry sloužící k odstranění nežádoucího šumu, což je stěžejním problém při segmentaci obrazu.

Použitá literatura a zdroje

[1] NIXON, Mark S. a Alberto S. AGUADO. *Feature Extraction & Image Processing* [online]. Academic Press. United Kingdom: Oxford, 2008 [cit. 2013-10-12]. second edition. ISBN 978-0-12372-538-7. Dostupné z: http://www.amazon.co.uk/Feature-Extraction-Image-Processing-Nixon/dp/0123725380

[2] GONZALES, Rafael a Richard WOODS. *Digital Image Processing*. third edition. Upper Saddle River, NJ.: Prentice Hall, 2008. 3. ISBN 9780131687288. Dostupné z:http://www.imageprocessingplace.com/DIP-3E/dip3e_main_page.htm

[3] HÁJOVSKÝ, Radovan, Radka PUSTKOVÁ a František KUTÁLEK. *Zpracování obrazu v měřící a řídící technice: učební text*. první. Ostrava: Ediční středisko VŠB – TUO, 2012. ISBN 978-80-248-2596-0.Dostupnéz:http://www.person.vsb.cz/archivcd/FEI/ZOMRT/Zpracovani%20obrazu%20v%20merici%20a%20ridici%20technice.pdf

[4] LINKA, Aleš, Petr VOLF a Miloslav KOŠEK. *Univerzitní e-learningový systém: Zpracování obrazu a jeho statistická analýza*[online]. 2004, 2004 [cit. 2013-10-12]. Dostupné z: http://e-learning.tul.cz/cgi-bin/elearning/elearning.fcgi?ID_tema=67&stranka=publ_tema

[5] SOJKA, Eduard. *Digitální zpracování a analýza obrazů* [online]. 1. vyd. Ostrava: VŠB - Technická univerzita, 2000, 133 s. [cit. 2013-10-12]. ISBN 80-707-8746-5. Dostupné z: http://mrl.cs.vsb.cz/people/sojka/dzo/digitalni_zpracovani_obrazu.pdf

[6] ŽÁRA, Jiří, Bedřich BENEŠ a Petr FELKEL. *Moderní počítačová grafika*. Vyd. 1. Praha: Computer Press, 1998, xvi, 448 s. ISBN 80-722-6049-9.

[7] ŠONKA, Milan, Václav HLAVÁČ a Roger BOYLE. *Image processing, analysis, and machine vision*. 3rd ed. Toronto: Thomson, 2008, xxv, 829 s. ISBN 978-0-495-08252-1.

[8] HLAVÁČ, Václav a Milan ŠONKA. *Počítačové vidění*. Praha: Grada, 1992, 272 s. ISBN 80-854-2467-3.

[9] ŠPANĚL, Michal a Vítězslav BERAN. *Obrazové segmentační techniky* [online]. Brno: Vysoké učení technické, 2005, 2006 [cit. 2013-10-12]. Dostupné z: http://www.fit.vutbr.cz/~spanel/segmentace/#_Toc125769325 . Přehled existujících metod

[10] HLAVÁČ, Václav. *Zpracování signálů a obrazů*. 1. vyd. Praha: Vydavatelství ČVUT, 2001, 220 s. ISBN 80-010-2114-9.

[11] PÁLKA, Zbyněk. Realizace hranového detektoru s využitím vlnkové transformace [online]. Brno, 2009 [cit.2013-10-13]. Dostupné z: https://dspace.vutbr.cz/bitstream/handle /11012/11674/Hranovy%20detektor.pdf?sequence=1. Bakalářská práce. Vysoké učení technické v Brně. Vedoucí práce Lukáš Růčka.

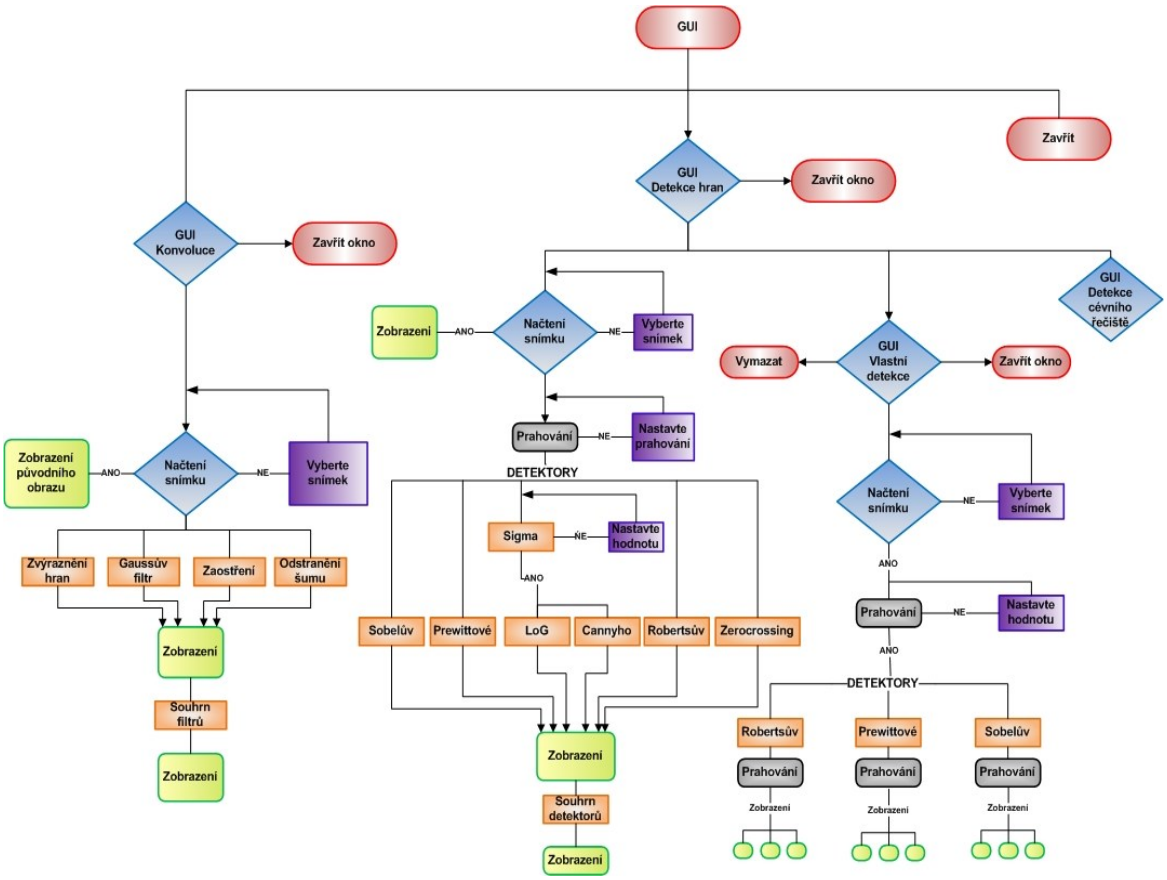
[12] ŽELEZNÝ, Miloš. *Zpracování digitalizovaného obrazu: učební text* [online]. Západočeská univerzita v Plzni, 2013, 2013 [cit. 2013-10-19]. ISBN verze souboru 130215. Dostupné z: http://www.kky.zcu.cz/uploads/courses/zdo/ZDO_aktual_130215.pdf. verze souboru 130215.

[13] ARENDT, Wolfgang a WARMA. Dirichlet and Neumann boundary conditions: What is in between?. In: *Dirichlet and Neumann boundary conditions: What is in between?* [online]. Basel, 2003 [cit. 2013-10-09]. DOI: 1424–3199/03/010119 – 17. Dostupné z: http://cantor.mathematik.uni-ulm.de/m5/arendt/publications/arendt-pub/short/2003-AreWar-DrcNmnBndCnd.pdf

[14] HORÁK, David. *Diskrétní transformace* [online]. Západočeská univerzita v Plzni, 2012 [cit. 2013-10-19]. Matematika pro inženýry 21. století. reg. č. CZ.1.07/2.2.00/07.0332. Dostupné z: http://mi21.vsb.cz/sites/mi21.vsb.cz/files/unit/diskretni_transformace.pdf .Ve spolupráci s VŠB - TUO.

Seznam příloh

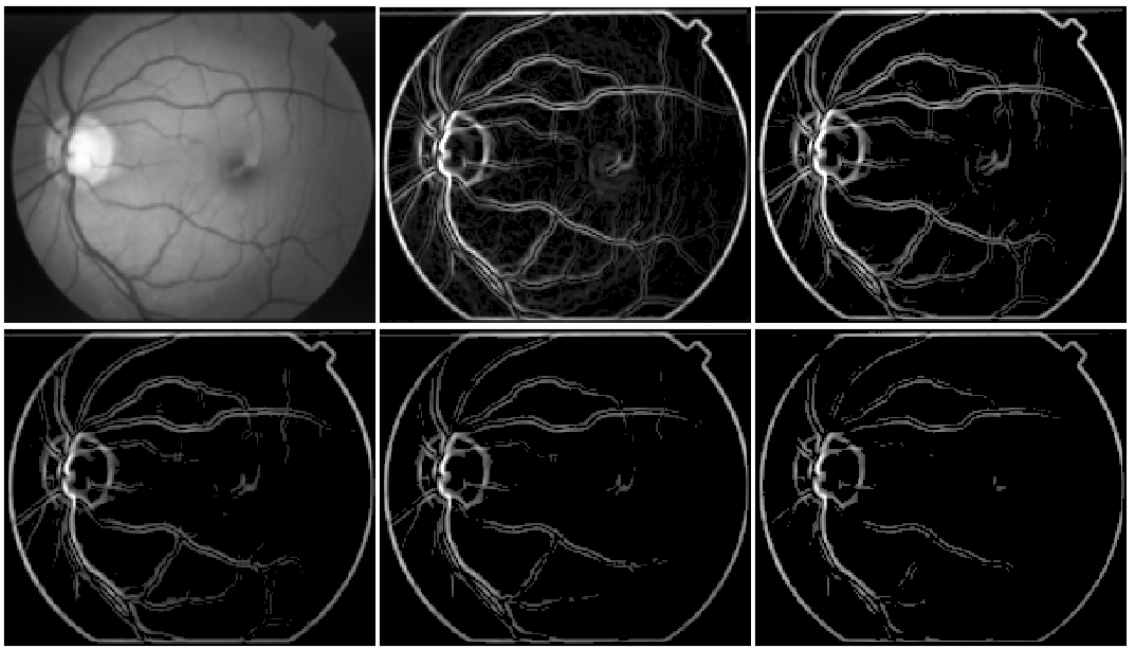
I. Vývojový diagram celkové aplikace GUI



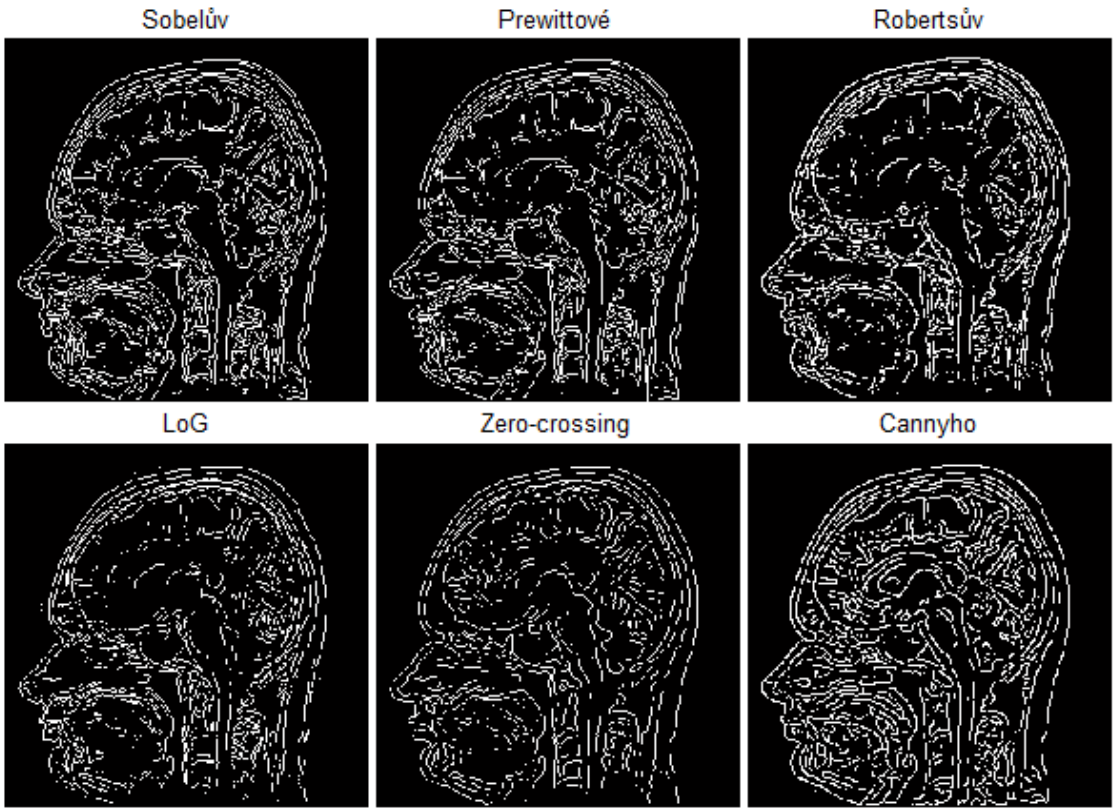
Obr. 68 Vývojový diagram popisující strukturu GUI vytvořeného v MATLABu.

II. Výsledné snímky po detekci hran

Obr. 69 Zleva: Původní snímek, implementovaný algoritmus s odlišným nastavením citlivosti prahu.



Obr. 70 Detekce hran implementovaným algoritmem – Sobelův detektor (citlivost práhu: 0,05; 0,10; 0,15; 0,20; 0,25).



Obr. 71 Souhrn dostupných hranových detektorů na snímek mozku.

Přílohy uložené na CD

K této práci je přiložené CD – ROM medium, které obsahuje následující soubory:

- Text této bakalářské práce ve formátu PDF/A a Microsoft Word 2010
- Zdrojové kódy
- Spustitelné GUI
- Použité medicínské snímky
- Help ke konvoluci a detekci hran